UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA EN INFORMÁTICA

PROYECTO FIN DE CARRERA

ESTUDIO Y DESARROLLO DE UN PROTOCOLO DE MOVILIDAD IPv6 LOCALIZADA BASADA EN LA RED

Autor: Ana Luz Cortés Vara
Tutor: Carlos Jesús Bernardos Cano

Febrero, 2011

Agradecimientos

Cuando llega el final de algo en lo que has trabajado mucho, nace siempre un sentimiento egoísta, por el cual te atribuyes todo el mérito, pero pensándolo detenidamente te das cuenta de que no es así, que no hubieras llegado al final, o al menos no con el mismo resultado, sin todas esas personas con las que compartes el día a día, y que directa o indirectamente han contribuido a que este momento llegue.

Entre esas personas, como no, se encuentra mi tutor, al que me gustaría agradecerle su paciencia y dedicación, porque no siempre ha sido fácil, pero sobre todo el que haya compartido conmigo parte de su conocimiento.

A toda mi familia, por estar siempre ahí, porque no importa lo que pase, siempre puedo contar con vosotros, no importa cuándo ni para qué os necesite que allí estaréis y os desvivís por intentar ayudarme y por hacer lo imposible para facilitarme todo al máximo.

Hoy, termina mi etapa universitaria, y aunque siempre entristece que algo bueno se acabe, sé que la gente que me ha acompañado a lo largo de estos años, seguirá estando conmigo, y al final, lo que hace un sitio especial es la gente con la que lo compartiste. Con vosotros he aprendido infinidad de cosas que no se enseñan en las aulas, hemos vivido momentos buenos y no tan buenos e incluso me habéis descubierto facetas de mi que ni si quiera yo sabía que existían. Pero lo más importante, sin lugar a dudas, es que esos momentos no se han quedado únicamente en la universidad.

Además de las personas nombradas, hay muchas más que siempre han estado incondicionalmente conmigo, procurando hacerme sentir bien y que simplemente con su presencia, muchas veces lo han conseguido.

Mención especial requiere una persona, que aunque ya no se encuentre aquí, siempre querré, por enseñarme cosas tan necesarias para vivir, porque sé que desde donde quiera que esté siempre me transmite su fuerza y alegría para seguir y me anima a afrontar todos los retos, y porque ahora mismo estará orgullosa de mí.

¡GRACIAS!

Alcance

Gracias a la tecnología existente, el acceso a Internet es posible desde diversos dispositivos, incluidos aquellos que pueden cambiar de posición dentro de la red fácilmente, como podría ser un teléfono móvil. Debido a ello, surge un nuevo problema; mantener la conexión a una red aunque la ubicación dentro de la misma varíe en el tiempo. Este concepto es lo que se denomina *Movilidad IP*. El movimiento de este tipo de terminales está estrechamente ligado a los enlaces inalámbricos, y cada movimiento de dicho terminal supondrá un cambio del punto de acceso a la red, tras el cual se espera que la conexión a la misma red se siga manteniendo. Para poder conseguir tal conectividad, en los últimos años se han desarrollado diversos protocolos que la facilitan, entre los que destacan los protocolos basados en la red. La característica más importante de este tipo de protocolos reside en que es la propia red la que gestiona la movilidad de los nodos que la forman.

Cuando se definió IPv4, no se contempló la posibilidad de escenarios en los que se pudiese llevar a cabo la movilidad de sus nodos. Por el contrario, en IPv6 sí que estuvo presente, lo que lo convierte en un protocolo mejor preparado para soportarla. Con lo cual, este proyecto estará basado en *movilidad IPv6*. Dentro de los protocolos existentes para IPv6, el protocolo objeto de estudio durante este proyecto es *Proxy Mobile IPv6* (PMIPv6).

El protocolo en el que PMIPv6 está basado, y que aún todavía sigue siendo motivo de investigación, es *Mobile IPv6* (MIPv6). Las ventajas principales que introduce este nuevo protocolo con respecto a su antecesor son; tras producirse un movimiento de un nodo, éste en su nueva ubicación, seguirá manteniendo la misma dirección IPv6 que tenía en su posición anterior (esto es posible debido a que cuando un nodo entra a formar parte de una red se le asigna un prefijo, el cual será mantenido durante todo el tiempo que permanezca dicho nodo en la red). Además, para que esto suceda, ese mismo nodo no necesitará disponer de ningún tipo de *software* adicional, con lo cual, dicho nodo podría no ser consciente de su cambio de ubicación dentro de la red.

A pesar de que ya ha sido definido, PMIPv6 aún está en una fase experimental, por lo que la finalidad de este proyecto ha sido conseguir una implementación que desarrolle correctamente los pasos básicos de dicho protocolo, en la cual se obviarán los aspectos más específicos del mismo, pero que garantizase, de manera robusta, que la comunicación de los diferentes elementos que componen la red fuese restablecida tras el cambio de posición de un nodo, tras un período de tiempo razonable. Por último, y una vez concluida la fase anterior, se llevará a cabo un plan de pruebas exhaustivo, para comprobar que, efectivamente, la implementación garantiza siempre la comunicación entre dichos elementos. Además se realizará una estimación del tiempo que la implementación emplea en conectar todos los elementos de la red nuevamente una vez producido un movimiento.

Abstract

Thanks to current technology, a wide variety of devices provide Internet connection, including mobile ones which could easily move around a network, such as a mobile telephone. Owing to this fact, a new difficulty emerges; keeping the connection between a device and the other elements belonging to the same network, despite the device changing its position within the network. This concept is called *IP Mobility*. This sort of movement is closely related to wireless links, and whenever the device moves, it will change the access point to the network. Nevertheless, the device must be able to communicate with other devices within the network. In order to obtain this connectivity, many protocols have been developed in the last few years. It is worth mentioning those protocols based on the network, in which the network is able to manage the mobility of the device, which have thus created the network.

Mobility had not been considered when IPv4 was defined, but it was taken into account during the IPv6 development. This is the reason to why this thesis has been focused on *IPv6 Mobility*. Specifically, the protocol which has been studied is *Proxy Mobile IPv6* (PMIPv6).

PMIPv6 is based on *Mobile IPv6* (MIPv6), which currently remains a research progress. There are two main advantages due to PMIPv6: when a device moves around the network, in its new position, the same IPv6 address will be assigned to that device (this is possible because when a node connects to a network, the network will apply a prefix to that device, which will remain with it until it leaves that network). Furthermore, the device does not require any additional software, so it may not have the awareness of its movement.

Although, PMIPv6 has already been defined, it is still in an experimental phase. For this reason, the main purpose of this thesis is to obtain a PMIPv6 implementation which is able to carry out the main features belonging to the protocol, guaranteeing a reliable behavior, i.e. the communication between all the nodes within a network will be reestablished after a reasonable period of time. Eventually, when the development is completed, it is necessary to prove that it works as expected. In order to do this the implementation is subjected to many control tests. In addition, a thorough study of the time, which is needed to recover the communication between the different elements belonging to the network, when a node changes its location, is outlined in the following report.

Todos nuestros sueños pueden convertirse en realidad si tenemos el coraje de perseguirlos.

Walt Disney (1901-1966)

Índice

Listado de Figuras

Listado de Tablas

Listado de Gráficas

CAPÍTULO

1

INTRODUCCIÓN

La madurez del hombre es haber vuelto a encontrar la seriedad con la que jugaba cuando era niño.

Friedrich Nietzsche (1844-1900)

1. 1 Marco general

En la sociedad en la que vivimos, denominada sociedad de la información, se ha convertido en una necesidad el estar comunicado permanentemente, y poder tener acceso a toda la información que se desee, en el momento que se desee. Esto se consigue conectando todos los ordenadores/nodos a través de una red, conocida como Internet. Lo cual nos lleva al problema de comunicación entre varios ordenadores. La tecnología actual posibilita la conexión a la red de un dispositivo móvil, el cual puede ir desplazándose, es decir, cambiar de ubicación, mientras sigue conectado a la red. Nace por tanto, el problema de la *movilidad IP*.

Concretamente, la *movilidad IP* pretende conseguir que un nodo continúe conectado a la red, siendo accesible por los demás nodos pertenecientes a ella, aunque éste cambie de punto de acceso a la misma. En el caso de que un nodo cambie su ubicación en la red, ya sea por un desplazamiento o por un cambio topológico de la red, puede suceder que éste quede incomunicado debido a su cambio de dirección IP, además de que dicho nodo, deberá ser capaz de crearse su nueva dirección.

Con el fin de resolver estos problemas, y poder brindar la libertad de movimiento a los nodos existentes a una red, se introduce la movilidad como parte del protocolo IPv6, añadida mediante nuevas cabeceras en los mensajes de dicho protocolo. El primer protocolo desarrollado para llevar a cabo esta función es Ipv6 móvil (Mobile IPv6, MIPv6) [4], el cual, lleva a cabo esta tarea introduciendo un nodo denominado *Home Agent* (HA) que mantenga una tabla que relacione al nodo que se desplaza con las diferentes direcciones IP que han sido utilizadas por él. De este modo, los mensajes dirigidos al nodo móvil (MN) siempre serán enviados al HA, y será él quien los redirija al MN buscando su nueva dirección IP en la tabla. Entonces, cuando un nodo cambie de ubicación, cambiará por tanto de dirección IP, que deberá creársela él mismo, y comunicársela al HA.

Otro protocolo que resuelve el mismo problema, y en el que se centra este trabajo es Proxy Mobile IPv6 (PMIPv6) [5]. Éste propone otro punto de vista de la movilidad, en la que el MN no realice ninguna acción, e incluso sea posible que ni el mismo sea consciente de su cambio de ubicación. Por tanto en el MN no se encuentre ningún tipo de software adicional para permanecer accesible tras un desplazamiento. PMIPv6 está basado en MIPv6, con lo cual puede reutilizar tanto las funciones del HA (*Local Mobility* Anchor, LMA en este nuevo protocolo) como el formato de los mensajes. Además, este nuevo protocolo introduce otro agente de movilidad intermedio (*Mobile Access* Gateway, MAG), el cual actuará como *proxy* para el MN. El MAG se convertirá en el *router* por defecto del MN, por lo tanto, todos los mensajes que este último envíe los recibirá el primero. El LMA mantendrá una responsabilidad parecida a la del HA en MIPv6, ya que todos los mensajes tanto destinados como enviados por el MN deberán pasar a través de él.

Este protocolo tiene dos principales ventajas que lo hacen más viable que el anterior: la primera y más importante, es que el MN no cambia de dirección IP al cambiar de ubicación, (por esta característica es por la que es posible que no sea consciente de su desplazamiento), ya que el MAG será el encargado de simular que siempre se encuentra en el mismo *link* de conexión a la red. La segunda es que no es necesario ningún SW adicional en al MN para que se pueda llevar a cabo este protocolo de manera satisfactoria.

Al inicio del proyecto, se trabajó con una implementación de PMIPv6 (todavía en desarrollo), creada inicialmente para MIPv6 y posteriormente ampliada para éste. Esta implementación es un único programa que toma como datos de entrada varios ficheros de configuración en los que se especifica el rol que va a desempeñar en el entorno la máquina que lo va ejecutar. Para un correcto funcionamiento del mismo se necesitará que el sistema operativo de los ordenadores cuente con una serie de características, que inicialmente están desactivadas.

Tras activar dichas características, e instalar el programa en los ordenadores necesarios, se procedió a una fase de validación de dicha implementación para certificar que efectivamente cumple con lo deseado de PMIPv6. Al no estar completa, ciertas especificaciones del protocolo no coincidían con un entorno de ejecución real, ya que era necesario forzar ciertas acciones para la realización del mismo. Entre ellas, la más destacada era la necesidad de que el MN siempre tuviera que enviar un *Router Solicitation* (RS) para que se le pudiese continuar siendo accesible aunque su punto de acceso a la red fuese distinto. Este mensaje es enviado cuando un nodo no conoce su *router* por defecto al que enviarle los mensajes.

Una vez completado el aumento de funcionalidad en la implementación se observó una gran inestabilidad en el comportamiento de PMIPv6, es decir, no siempre se obtenía el resultado previsto del mismo. Por lo tanto dicha implementación no sería viable, debido a que, como se ha mencionado anteriormente, cuando se ofrece un servicio, se espera que éste sea cumplido. Por tanto, se decidió abordar el problema de la movilidad IP desde el principio, siendo solucionado igualmente mediante PMIPv6.

Esto lleva a una implementación nueva de PMIPv6. Dado su gran tamaño, no se han desarrollado todas sus funcionalidades, pero se ha conseguido que al menos, respetando toda la señalización requerida por el protocolo, el MN cambie de punto de acceso a la red y siga siendo accesible por cualquier otro nodo, manteniendo la misma dirección IPv6 (característica imprescindible en el protocolo mencionado). En este caso, la implementación constará de dos programas diferentes, siendo ejecutado cada uno de ellos en el ordenador que vaya a asumir el rol para el que está destinado cada uno de los programas.

Además, en el presente documento se incluye un estudio detallado sobre el tiempo que es necesario para restablecer la comunicación con el MN una vez que éste haya cambiado su ubicación dentro de la red. Obviamente, esta comunicación queda interrumpida debido a que primeramente se desconecta de un punto de acceso para a continuación conectarse con

otro. Este nuevo punto de acceso, deberá repetir toda la señalización con el LMA correspondiente, es entonces y sólo entonces, cuando se restablecerá dicha comunicación.

1. 2 Objetivos

Como objetivo principal, el proyecto se centra en conseguir una implementación realista de un protocolo de movilidad IPv6. En este caso, el seleccionado es PMIPv6, ya que se considera el más viable y para poder ser implantado. Por ello inicialmente, como primer propósito, se realiza un estudio detallado sobre su funcionamiento [5], así como sobre las bases teóricas del mismo [2].

Para alcanzar dicho objetivo, primeramente se parte de una implementación de PMIPv6 realizada por varias entidades. Dicha versión, inicialmente fue creada para desarrollar el protocolo MIPv6, y posteriormente fue incrementada para que llevase a cabo el protocolo objeto de estudio de este proyecto.

Para ponerla en funcionamiento, es necesaria la utilización de al menos 4 equipos. Dos que actúen como MAG (para llevar a cabo un cambio de punto de acceso), otro que ejerza como LMA, y por último el MN, siendo éste un ordenador portátil. Además, para poder desarrollar todas las funcionalidades de PMIPv6 es imprescindible preparar los ordenadores que actúen tanto de MAGs cómo de LMA. Es entonces cuando se procede a la recompilación del *kernel* de todos ellos para añadirle determinadas opciones.

Una vez realizada la operación anterior, ya se puede ejecutar la implementación. Es entonces, cuando se observan ciertas deficiencias en la misma, por lo que el objetivo se encamina a intentar solventarlas. Es en una de ellas, la más importante, ya que requiere un cambio del código fuente, al intentar incrementar la funcionalidad del mismo, donde se observa la inconsistencia de la implementación utilizada.

Por lo tanto, y siendo fieles al objetivo principal, se decide comenzar una nueva implementación de PMIPv6 que cumpla con los requisitos deseados. Si bien, esta nueva implementación no llevará a cabo todas las funcionalidades de PMIPv6, debido a su elevado número. Pero, como punto positivo a destacar, se logrará una gran robustez en él, al conseguir restablecer la comunicación entre el MN y los demás elementos de la red en todas las circunstancias. Esta nueva implementación, al igual que la anterior, es desarrollada para un entorno Linux, y se usa como lenguaje de programación, el lenguaje C.

Una vez concluida con éxito la fase anterior, se decide realizar un estudio detallado sobre el tiempo en el que el MN es inaccesible por otros nodos de la red. Esto, sucede cuando el nodo cambia de punto de acceso a la red. Durante este período, el MN no es capaz ni de enviar ni de recibir paquetes, lo cual provoca pérdida de mensajes y retardo en los mismos.

Este problema se considera crítico en aplicaciones que requieran ser ejecutadas en tiempo real. Por esto mismo, se considera interesante, ya que dependiendo del tiempo que sea necesario para ello, este protocolo podrá o no, ser utilizado para determinadas aplicaciones.

1. 3 Fases del desarrollo

El desarrollo de este proyecto puede ser dividido en cinco grandes fases. Como se puede observar, en el anexo B, estas fases serán divididas en tareas para obtener un diagrama detallado de las actividades realizadas en él. Estas fases generales coinciden con:

- **Documentación:** como primera instancia, y antes de empezar con el proyecto en sí, es necesario adquirir una serie de conocimientos previos, así como un estudio detallado del protocolo PMIPv6, en el cual se basa este proyecto.

- **Instalación de la primera implementación:** a continuación, se procede a realizar todas las operaciones necesarias para poder poner en funcionamiento la primera implementación de PMIPv6 que fue empleada en el proyecto.

- **Mejora de las funcionalidades:** el siguiente paso llevado a cabo fue el intento de incrementar las funcionalidades de esta implementación para que su utilización se asemejase más a la realidad. En esta fase, también será necesario la preparación de ciertos dispositivos, así como una validación del mismo incremento, que nunca pudo concluirse.

- **Implementación de PMIPv6:** se llega a esta fase debido a los diversos problemas ocurridos en la fase anterior. En ella se realizaran todas las tareas necesarias para el desarrollo de un SW, como son: análisis, diseño, etc.

- **Evaluación del funcionamiento:** completada la fase anterior, se procede a validar la aplicación realizada. Además, se realiza un estudio estadístico detallado del tiempo empleado desde la pérdida de comunicación hasta el restablecimiento de la misma.

- **Escritura de la documentación del proyecto:** las últimas tareas que deben ser realizadas será la escritura del presente documento, así como la preparación de la presentación del proyecto ante el tribunal.

1. 4 Estructura del documento

El presente documento está dividido en 7 capítulos, entre los cuales se incluye uno que contendrá todos los anexos pertenecientes al mismo. Éstos son descritos a continuación:

- **Capítulo 1**, "Introducción". Presenta una visión global del proyecto realizado, ya que en él está incluido un breve resumen, presentado en el "marco general", los objetivos del mismo, así como las fases que han sido necesarias para su realización.
- **Capítulo 2**, "Definiciones y acrónimos". En él, se realizara una descripción de varios términos importantes para la comprensión del proyecto. Además, como su propio nombre indica, también se ha incluido una lista de los acrónimos utilizados a lo largo del documento.
- **Capítulo 3**, "Estado del arte". En este capítulo se introducirán los conocimientos previos necesarios para un correcto entendimiento de la tarea realizada en el proyecto. Entre otros, se incluye nociones básicas de redes de ordenadores, así como el protocolo IPv6, y el concepto de *movilidad IP*, sin olvidar el protocolo *PMIPv6*, principal objeto de estudio.
- **Capítulo 4**, "Trabajo realizado". Es en este capítulo donde se detallará todo lo relativo al desarrollo de una implementación del protocolo PMIPv6, tras haber intentado aumentar la funcionalidad de otra implementación existente.
- **Capítulo 5**, "Validación y evaluación". Introduce el escenario en el cual se han realizado las pruebas para validar su funcionamiento, también expuestas en este capítulo. Además, en él, se realiza un estudio estadístico detallado del tiempo que incomunicación del MN tras haber cambiado de punto de acceso a la red.
- **Capítulo 6**, "Conclusiones". Se especifican todas las conclusiones obtenidas tras la realización del proyecto, así como trabajos futuros de investigación posteriores a él.
- **Capitulo 7**, "Anexos". Se presentan todos los anexos referenciados a lo largo de este proyecto. Entre ellos se incluyen todos los detalles sobre la realización del mismo. Estos son:
    A. Funcionamiento detallado de PMIPv6: este anexo será una continuación del apartado *estado del arte* en el que se expondrá con un nivel mayor de detalle el funcionamiento del protocolo estudiado en este proyecto.
    B. Gestión del proyecto: en él se encontrará las diferentes fases en las que ha sido dividido el proyecto, así como su duración y un presupuesto que estima los costes del mismo.

C. Preparación del entorno de pruebas: en este anexo se describirá con detalle todas las operaciones que han sido necesarias para poder ejecutar PMIPv6 en el entorno de pruebas propuesto.
D. Recompilación del *kernel*: aquí, se describirán detalladamente todos los pasos necesarios para recompilarar el *kernel* de un equipo, añadiéndole las características deseadas.
E. Cross Compiling: en este anexo, se detallará la forma de llevar a cabo este proceso para poder instalar un programa en un *router* de los empleados en el desarrollo del proyecto.
F. Guía de usuario: será en este anexo dónde se describa las acciones necesarias que un usuario puede realizar para ejecutar y configurar el programa implementado.
G. Trabajo con otra implementación: describirá todo el trabajo que se realizó tomando como base la primera implementación de PMIPv6 con la que se trabajo intentando llegar al objetivo propuesto.
H. Tabla t-STUDENT: tabla t-student, dónde se obtuvieron determinados datos para el análisis estadístico presentado en el capítulo 5.
I. Listado de referencias: en él se encontrarán todas las referencias citadas a lo largo de este documento, y necesitadas tanto para la creación del mismo como para el desarrollo del programa.

CAPÍTULO

2

DEFINICIONES Y ACRÓNIMOS

La mayoría de las ideas fundamentales de la ciencia son esencialmente sencillas y, por regla general pueden ser expresadas en un lenguaje comprensible para todos.

Albert Einstein (1879-1955)

2. 1 Definiciones

- **Acceso global:** término ligado a dirección IP. En el caso en el que una dirección IP sea de acceso global, entonces denotará que se podrá alcanzar la interfaz a la que se le asignó tal dirección desde fuera de la red en la que se encuentra, mediante dicha dirección.

- **Acceso local:** al contrario de la definición anterior, en este caso únicamente se podrá acceder a una dirección IP de acceso local en el caso en el que el nodo que desea comunicarse con dicha interfaz se encuentre dentro de la misma subred que ella.

- **Acción:** Ejercicio de la posibilidad de hacer. Resultado de hacer [1].

- **Agente de movilidad:** Cada uno de los elementos pertenecientes al entorno de pruebas, que ejecuta alguna acción para hacer viable la movilidad de algún otro elemento del mismo entorno de pruebas.

- **Arquitectura:** estructura física y lógica de los componentes de un ordenador.

- **Backoff exponencial:** algoritmo utilizado cuando se deben realizar retransmisiones de mensajes en una red de ordenadores que consiste en que dichas retransmisiones serán enviadas en franjas de tiempo cada vez más largas, dependiendo de un cierto criterio.

- **Checksum:** suma de redundancia con la finalidad de protección de los datos que son incluidos en el mensaje en el que se encuentra.

- **Comando:** mandato, instrucción u orden que el usuario proporciona a un ordenador para que éste lo ejecute.

- **Compilar:** proceso mediante el cual un código fuente, escrito en un lenguaje de alto nivel, es traducido a un código que una máquina sea capaz de ejecutar.

- **Comunicación:** Según [1], transmisión de señales mediante un código común al emisor y al receptor. En este documento cuando se hable de este término se referirá a la capacidad de cada uno de los nodos para poder enviar información a otros nodos y que ésta sea recibida.

- **Conexión:** enlace, atadura, concatenación de un elemento con otro. Punto donde se realiza el enlace entre elementos.

- **Debian:** nombre de una distribución del sistema operativo Linux, en la cual está basada la distribución que se ha utilizado para el desarrollo de este proyecto (Ubuntu).

- **Detached:** independiente, separado. En el documento, se encontrará este término ligado al concepto *thread*, el cual indicará que será ejecutado con total independencia a los demás *threads*, no necesitando de ellos ninguna información, ni teniendo que esperar a que alguno de ellos termine para poder ser ejecutado.

- **Dirección IP:** etiqueta numérica asignada a cada dispositivo de una red, con el fin de que puedan ser identificados de una manera lógica y jerárquica dentro del protocolo IP. A cada interfaz de cada dispositivo le corresponderá una.

- **Dispositivo:** mecanismo dispuesto para realizar o producir una acción prevista.

- **Elemento:** Cada uno de los componentes de un conjunto. En el presente documento, el conjunto será el entorno de pruebas y los elementos corresponderán los dispositivos que en él se encuentren.

- **Encaminar:** sinónimo de enrutar.

- **Encapsular:** colocar los datos que van a ser enviados por una red en paquetes que se puedan administrar y rastrear.

- **Entrada:** cada uno de los elementos pertenecientes a una lista.

- **Enrutar:** enviar un paquete por el camino adecuado para que éste llegue a su destino.

- **Firma RSA:** sistema criptográfico de clave pública, mediante el cual se garantiza tanto el contenido del mensaje como la identidad de quien lo envía.

- **Firmware:** bloque de instrucciones de programa con una finalidad específica, almacenado en memoria no volátil, y que estable la lógica de más bajo nivel, controlando los circuitos electrónicos del dispositivo en el que se encuentre.

- **Gestionar:** realizar las operaciones que sean necesarias para lograr el fin deseado.

- **Handoff:** procedimiento mediante el cual se transfiere el servicio facilitado por una entidad a otra entidad del mismo tipo.

- **Host:** sinónimo de elemento.

- **Implementación:** sinónimo de programa.

- **Interfaz:** conexión física y funcional entre dos dispositivos independientes que les permite comunicarse a distintos niveles.

- **Kernel:** núcleo de un sistema operativo. Es el encargado de garantizar un acceso seguro y de gestionar a los recursos HW del ordenador donde se esté ejecutando.

- **Levantar:** en este documento con el término levantar, siempre seguido de interfaz, coincidirá con activar.

- **MACRO:** serie de instrucciones que se almacenan para que puedan ser ejecutadas de forma secuencial mediante una única orden de ejecución.

- **Makefile:** archivo que posibilita la compilación y la creación del ejecutable de un programa mediante un único comando.

- **Máquina:** sinónimo de ordenador.

- **Mensaje:** unidad fundamental de transporte de información en una red de ordenadores. En él se incluirán tanto todas las cabeceras necesarias para que el mismo pueda ser enviado, como la información que se considere necesaria.

- **Movilidad:** cualidad mediante la cual un elemento puede cambiar de ubicación.

- **Nodo:** sinónimo de elemento.

- **Operación:** sinónimo de acción.

- **Ordenador:** posible elemento del conjunto de pruebas. Máquina electrónica dotada de una memoria de gran capacidad y de métodos de tratamiento de la información, capaz de resolver problemas aritméticos y lógicos gracias a la utilización automática de programas registrados en ella [1].

- **Parsear:** proceso mediante el cual se analiza una secuencia de símbolos con el fin de extraer de ella la información que se considere oportuna.

- **Path:** conjunto ordenado de todos los directorios a los que se debe acceder para llegar a un archivo determinado.

- **Payload:** datos incluidos en las cabeceras de los mensajes enviados a través de una red de ordenadores.

- **Perfil:** conjunto de rasgos peculiares que caracterizan a un elemento.

- **Permiso:** licencia o consentimiento para realizar una acción.

- **Prefijo:** en el caso concreto de este documento, se denominará prefijo a la secuencia de números que definirán una red. Por lo tanto, toda dirección IP perteneciente a dicha red deberá comenzar por la secuencia que le haya sido asignada a dicha red.

- **Privilegio:** sinónimo de permiso.

- **Programa:** secuencia de instrucciones que un dispositivo es capaz de interpretar y ejecutar, y que le permite realizar acciones diversas.

- **Proxy:** dispositivo que realiza una acción en representación de otro.

- **Punto de acceso:** dispositivo que conecta otros equipos para formar una red. En el caso de este proyecto, hará de intermediario o puente entre los equipos *wifi* (nodos móviles) y la red Ethernet.

- **Registrar:** Inscribir, dar de alta un elemento en una lista.

- **Red:** conjunto de equipos informáticos conectados entre sí que pueden intercambiar información.

- **Rol:** función que algo o alguien cumple [1]. Conjunto de patrones de conducta esperado por algo o alguien que ocupa una posición determinada.

- ***Router*:** dispositivo para la interconexión de redes de ordenadores que opera en el nivel tres de la torre OSI, y permite asegurar el enrutamiento de paquetes entre redes, o determinar el mejor camino que deben tomar los mismos. Además, en este documento se considerará como el dispositivo que da acceso al entorno de pruebas creado, es decir a la red.

- **Ruta:** conjunto de todos los nodos de una red por los que pasa un mensaje desde el nodo que lo envío hasta la dirección destino del mismo.

- **Sello temporal:** marca de tiempo que indicará el momento de la creación del elemento o entrada que lo contenga.

- **Semáforo:** mecanismo de sincronización para el acceso, por parte de los diferentes hilos de ejecución, a los recursos compartidos de un mismo programa.

- **Señalización:** conjunto de mensajes intercambiados entre los diferentes nodos de una red para conseguir un fin común.

- **Shell:** interfaz utilizada por el usuario para introducir comandos que deban ser ejecutados por el núcleo del sistema operativo.

- **Sniffer:** tipo de programa que captura todos los paquetes tanto enviados como recibidos por la interfaz en la que está siendo ejecutado.

- ***Socket*:** conector mediante el cual dos dispositivos pueden intercambiar mensajes. Se debe crear uno en cada elemento del mismo tipo para que la comunicación se produzca de forma satisfactoria. En el caso de un *socket pasivo* su única misión será estar esperando continuamente la llegada de algún mensaje para su posterior procesamiento.

- **SSH:** protocolo mediante el cual se puede acceder a máquinas remotas a través de una red.

- **Switch:** conmutador que conecta dos o más segmentos de una misma red.

- **Thread:** hilo de ejecución de un programa.

- **Traza:** marca dejada durante la ejecución de un programa cuando cambia de posición, es decir, secuencia de acciones ejecutadas por dicho programa que quedan reflejadas, ya sea en la terminal/*Shell* dónde está siendo ejecutada o en un fichero.

- **Túnel:** canal de comunicación directa entre dos nodos.

- **Tunneling:** procedimiento por el cual se encapsula un mensaje de un protocolo de red sobre otro, creando un túnel de comunicación entre los dos extremos que se quieran comunicar.

- **Usuario root:** usuario que puede realizar cualquier acción en un ordenador ya que dispone de todos los privilegios posibles.

- **Vecino:** un vecino de un determinado nodo, es un nodo que se encuentra directamente conectado al primero.

- **Wireless:** WIFI. Sin cables. Comunicación inalámbrica, es decir, los dos extremos que establecen la comunicación no están conectados por un medio de propagación físico, sino que se utiliza la modulación de ondas electromagnéticas a través del espacio.

2. 2 Acrónimos

- **BC:** Binding cache.
- **BUL:** Binding Update List.
- **C.C:** Cross Compiling.
- **CN:** Correspondent Node.
- **CoA:** Care of Address.
- **CVS:** Concurrent Version System.
- **DAD:** Duplicate Address Detection.
- **DHAAD:** Dynamic Home Agent Address Discovery.
- **DHCP:** Dynamic Host Configuration Protocol.
- **DNS:** Domain Name Server.
- **DoS:** Denegation of Service.
- **Fmw:** Firmware.
- **HA:** Home Agent.
- **HW:** Hard Ware.
- **ICMPv6:** Internet Control Messages Protocol version 6.
- **ID:** Identification.
- **IKEv2:** Internet Key Exchange version 2.
- **IPsec:** Internet Protocol Security.
- **IPv6:** Internet Protocol version 6.
- **ISP:** Internet Server Provider.
- **LMA:** Local Mobility Anchor.
- **MAC:** Media Access Control.
- **MAG:** Mobile Access Gateway.
- **MH:** Mobility Header.
- **MIPv6:** Mobile IP version 6.
- **MN:** Mobile Node.
- **MTU:** Maximum Transfer Unit.
- **NA:** Neighbor Advertisement.
- **NS:** Neighbor Solicitation.
- **PBA:** Proxy Binding Acknowledgement.
- **PBU:** Proxy Binding Update.
- **PMIPv6:** Proxy Mobile IP version 6.
- **RA:** *Router* Advertisement.
- **RS:** *Router* Solicitation.
- **RSA:** Rivest, Shadmir, Adleman.
- **RTT:** Round Trip Time.
- **SDK:** Software Development Kit.

- **SSH:** Secure Shell.
- **SW:** SoftWare.
- **TCP:** Transmission Control Protocol.

CAPÍTULO

3

ESTADO DEL ARTE

Daría todo lo que sé por la mitad de lo que ignoro.
René Descartes (1596-1650)

3. 1 Introducción

La comunicación entre ordenadores (como puede ser a través de Internet) es un problema multidisciplinar cuyo grado de complicación es bastante elevado. Para poder resolverlo, se divide en tareas más fáciles de resolver y se agrupan en las denominadas *capas* o *niveles*. Estas *capas* se unen formando una torre *fig.* 1. Cada capa da soporte a la capa inmediatamente superior, facilitándole la tarea a desarrollar, siendo independientes unas de otras. Obviamente, una *capa* sólo podrá comunicarse con su semejante en otros ordenadores. El modelo utilizado actualmente, TCP/IP, consta de 5 niveles [2]:

- **Nivel físico:** envía y recibe bits a través de una interface.
- **Nivel de enlace:** controla el acceso al nivel físico, corrige y detecta errores, y en algunos casos proporciona seguridad a la red.
- **Nivel de red:** es el responsable de enviar los paquetes de un nodo a otro.
- **Nivel de transporte:** controla el flujo de datos enviados entre los dos dispositivos. Dentro de esta *capa* se encuentran los protocolos TCP, el cual proporciona una transmisión fiable de los paquetes, y UDP, que no garantiza la recepción de todos los paquetes (usado en aplicaciones de tiempo real).
- **Nivel de aplicación:** toma las entradas del usuario, y las formatea antes de enviárselas a las capas inferiores.

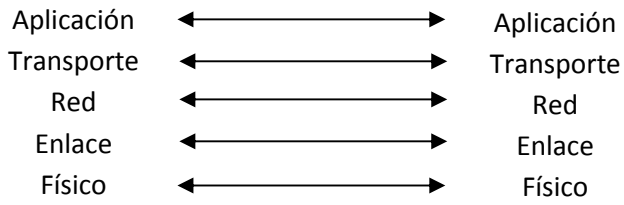


Figura 1: Torre TCP/IP

Un dispositivo no podría ser conectado a todos los que hay en su misma red, ya que a medida que la red creciese, el número de conexiones crecería con él, y la red llegaría a ser inviable. Un *router* es un nodo que puede conectar un dispositivo con otro. Gracias a ellos, dos nodos que no estén en la misma interfaz (entendiendo por interfaz el mismo punto de unión, como podría ser un cable) pueden establecer una comunicación. Un nodo con más de una interfaz es llamado nodo *multihomed*. Por definición, los *router*s suelen ser *multihomed*, pero los nodos también pueden serlo. La diferencia entre ellos es que el *router* recibe paquetes que no están destinados a él para que los encamine, mientras que los nodos (*hosts*) únicamente recibe los paquetes destinados a él.

Los *router*s usan direcciones IP para localizar nodos en internet, y así poderle enviar paquetes. Las direcciones de IPv4 constan de 32 bits, mientras que las de IPv6 de 128bits. Estas direcciones no son permanentes, pueden cambiar por numerosos motivos, como puede ser una reorganización de la red, unir varias redes etc. Además de las direcciones IP, a cada nodo en la red se le asigna un nombre, que suele ser el que se conoce a la hora de intentar comunicarse con dicho nodo. Para conseguir la dirección IP conociendo el nombre o viceversa, existen los denominados DNS. Un servidor DNS es una base de datos distribuida que almacena la equivalencia entre nombres y direcciones IP. La búsqueda se realiza en árboles con jerarquía, así pues cada rama del árbol tiene asignado un dominio (org, com, es, de, …). Cada nodo tiene asociado un nombre único (al igual que las direcciones IP), aunque el mismo nombre se puede repetir dentro de dominios distintos. Se sugiere siempre asegurar la conexión a algún servidor DNS, por lo que normalmente se está conectado a dos, uno primario y un secundario, para conseguir redundancia, y así evitar posibles fallos. La respuesta de un servidor DNS suele almacenarse en la memoria *Cache* para evitar estar consultando frecuentemente la misma dirección IP.

Un *socket* es un registro que incluye información sobre la conexión entre dos aplicaciones. La información sobre la aplicación es representada por el número de puerto. Ciertas aplicaciones tienen un número de puerto reservado, como puede ser HTTP con el puerto 80, y otras que no tienen ninguno asociado, se le fija uno efímero para poder crear el *socket*, y que el nivel de transporte sepa por qué aplicación va a ser usado. Un *socket* guardará los parámetros necesarios para que una comunicación pueda llevarse a cabo: dirección IP del emisor, dirección IP del receptor, número del protocolo de transporte, número de puerto de la aplicación emisora, número de puerto de la aplicación receptora. Si alguno de estos parámetros es modificado durante la comunicación, la conexión quedará interrumpida y será necesario volver a establecerla.

Una vez establecido lo anterior, y para que un dispositivo pueda comunicarse con otro se enviarán paquetes entre ellos. Los paquetes los generará la capa de aplicación con la información que desee mandar, y se enviará al nivel inmediatamente inferior, éste le añadirá su cabecera para poder comunicarse con la misma capa del otro dispositivo, y lo pasara el nivel inferior, y así sucesivamente hasta pasar por todos los niveles y enviar el paquete. Este proceso es denominado *encapsulación*. El proceso contrario, *demultiplexación*, se produce cuando un paquete llega a un nodo, en este caso, el primer nivel que pasará será el nivel físico, éste le quitará su cabecera al paquete y lo pasará al nivel de enlace, que le quitará su cabecera y lo enviará al nivel superior, y así hasta que llegue a la capa de aplicación.

El encaminamiento en una red es la manera en la que se envían los paquetes desde el emisor hasta el receptor. Siempre se intentan encaminar por el mejor camino, teniendo en cuenta cual será el siguiente salto. Una configuración manual para saber que *router*s están conectados entre sí o para saber el *router* al que está conectado cada nodo no sería factible debido a las dimensiones que pueden llegar a alcanzar las redes, por lo que se suele utilizar protocolos de encaminamiento entre *router*s, consiguiéndose de esta manera que cada *router* informe de los links, o nodos que es capaz de alcanzar, y del coste que conlleva alcanzarlos.

Para poder encaminar todos los paquetes que se envían y debido a la gran cantidad de nodos existentes en Internet, es importante agrupar las rutas según los *router*s por los que sea necesario pasar. De esta forma se obtendrán conjuntos de *host*s, que se sabe que son alcanzados por un determinado *router*, y a ese determinado *router* se llega por otro, y así sucesivamente, esto es lo que se conoce como agregación. Esto se puede conseguir si se distribuyen las direcciones IP de un ISP (Internet Service Provider) en bloques jerárquicos. De esta forma, la dirección IP puede ser dividida en dos, el prefijo, que identificará la red, y el resto de la dirección identificará al nodo. Por lo tanto, podrá conocer todas las direcciones IP pertenecientes a una determinada red, y que esa red es alcanzable a través de un/os determinado/s *router*/s.

3. 1. 1 IPv6

Debido al gran incremento del número de nodos en Internet, tanto *host*s como *router*s, las direcciones que IPv4 puede albergar (alrededor de 4 billones) son insuficientes. Por lo tanto, se necesita un formato de direcciones IP capaz de albergar más nodos, es decir, un mayor número de bits que definan dichas direcciones. Mientras que en IPv4 se utilizaban 32 bits para especificar una dirección IP, en IPv6 se utilizaran 128bits, con lo cual el número de nodos que podrán ser direccionados incrementan de forma considerable. Con este incremento, se consigue además; una simplificación en el procedimiento de autoconfiguración de direcciones IP en los nodos, reducir el número de entradas en los *router*s, y poder incluir seguridad en la *capa* de red. Esta nueva versión, además de evitar muchos de los problemas asociados con IPv4, fue diseñada para facilitar la movilidad de los dispositivos en la red.

El protocolo IPv6 está especificado en el RFC 2460 [3]. En él, se detalla el significado de las distintas cabeceras con sus respectivos campos y opciones, y los posibles valores que pueden tomar. Para simplificar el tratamiento de los paquetes IP, tanto en los nodos intermedios como en el nodo final, se ha fijado el tamaño de las cabeceras en 40 octetos, las cabeceras de extensión, u opciones no tendrán tamaño máximo. Todo ello alineado a 64 bits. IPv6 no contiene el campo *Checksum*, dado que dicho campo fue diseñado para asegurar que el contenido de los paquetes no era modificado a lo largo de su recorrido por la red, y eso, es poco probable que suceda. Además, protocolos de niveles superiores tienen mecanismos tanto para detectar como para corregir errores, por lo que dicho campo no es necesario.

Además de lo comentado anteriormente, IPv6 añade más cabeceras y opciones para mejorar la comunicación entre dos dispositivos. El encaminamiento en IPv6 es bastante similar al utilizado en IPv4, ya que los protocolos utilizados en la versión 4 sufren muy pocas variaciones en la versión 6. Los únicos cambios realizados en los mismos tienen que ver con el formato, para adaptarlos al formato IPv6. Para realizar el encaminamiento de un paquete, se podrán utilizar nuevas cabeceras añadidas para tal efecto. Cada una de las cabeceras incluidas

en el paquete, cabecera comenzará con un campo de 8 bits que indicará el tipo de la siguiente cabecera. Las principales cabeceras utilizadas en IPv6 son:

- **Cabecera de opciones de salto a salto:** incluye las opciones que necesitan ser procesadas por cada nodo a lo largo del recorrido del paquete en la red, esto es tanto los intermedios como el nodo destino. Puede ser usada por ejemplo en cualquier aplicación que necesite un estado de instalación en todos los nodos del camino.
- **Cabecera de encaminamiento:** especifica el camino a seguir por los paquetes desde un determinado origen hasta un destino. Se indica detallando el número de nodos (incluyendo el final) por los que un paquete debe pasar. La dirección IPv6 destino que contenga el paquete será el siguiente nodo que el paquete vaya a visitar camino del destino. Los demás nodos del camino estarán incluidos en esta cabecera. Cada vez que pasa por un nodo, éste pondrá su dirección al final de la lista de direcciones de la cabecera y reducirá en uno el número de nodos por los que tiene que pasar el paquete, por lo tanto cuando llegue al nodo destino, éste número será cero.
- **Cabecera de fragmentación:** la capa de enlace soporta una determinada cantidad de datos, si la información que se desea enviar en un paquete es superior a esa cantidad, dicha información ha de ser fragmentada y encapsulada en varios paquetes, cuyo tamaño sea soportado por el nivel de enlace. Si se pasa por más de un enlace, el tamaño máximo de los paquetes será el mínimo tamaño que soporten todos los enlaces. La información fragmentada será reconstruida en el destino antes de ser enviada al nivel de transporte. Cada fragmento (paquete) contendrá la cabecera de fragmentación. Un campo importante de esta cabecera es el de identificación puesto que todos los fragmentos deberán portar el mismo identificador, que será inequívoco, para reconocer los paquetes pertenecientes a una misma información.
- **Cabecera de autenticación:** es un mecanismo definido para proteger la integridad del paquete a lo largo de su recorrido por la red. El emisor debe calcular el *checksum* del paquete entero, teniendo en cuenta los campos que no van a variar a lo largo del recorrido (inmutables) y el valor predecible de los campos que sí que variaran. Las políticas de seguridad son las que decidirán qué paquetes y de qué manera deberán ser protegidos.
- **Cabecera de encriptación:** con esta cabecera se pretende conseguir confidencialidad, es decir, que ningún nodo excepto el destino y el origen entenderán la información enviada. Para que este mecanismo pueda funcionar, ambos extremos (origen y destino), deben haberse puesto previamente de acuerdo para conocer el método por el que se van a proteger los paquetes pertenecientes a dicha comunicación. Evidentemente, las cabeceras no serán encriptadas, ya que en el caso de serlo los *router*s intermedios no las entenderían y por lo tanto no podrían encaminar los paquetes.

- **Cabecera de las opciones de destino:** contiene las opciones que deberían ser tratadas por todos aquellos nodos que se encuentren en el campo de dirección destino. Esta cabecera es parecida a la de opciones de salto a salto, la diferencia es que en este caso es el procesamiento que tienen que hacer los nodos que son destino, mientras que en el otro, es todos los nodos a lo largo del camino.

El RFC 2460 [3], recomienda seguir un orden de las cabeceras. Dicho orden es el mismo en el que se han expuesto las cabeceras en este documento.

ICMPv6, al igual que en IPv4, es utilizado para enviar información de control entre los dos extremos de la comunicación. Habitualmente se suelen mandar mensajes de error, es decir que algo ha fallado en la comunicación entre ellos. Estos mensajes pueden ser del tipo de destino inalcanzable, paquete demasiado grande, tiempo de espera o envío superado, o cualquier problema en algún parámetro.

Tunneling es el proceso por el que un nodo (*host* o *router*), encapsula un paquete IPv6 en otra cabecera IPv6, por lo tanto el paquete resultante contendrá más de una cabecera IPv6. Es usado, por ejemplo en Mobile IPv6.

Las direcciones en IPv6 son asignadas a interfaces, no a nodos, ya que un nodo puede tener más de una interfaz física. Una dirección IPv6 puede ser dividida en dos partes fundamentales, el prefijo, que define el número de bits que son asignados a un determinado link, y el identificador de la interface, que como su propio nombre indica, define la interface a la que se le ha asignado dicha dirección. Existen cinco tipos de direcciones en IPv6.

- **Direcciones *Unicast*:** son asignadas a una única interfaz. Típicamente cada dispositivo en Internet tiene asignada una dirección de este tipo. Existen cuatro ámbitos en los que estas direcciones pueden existir: dentro de un nodo (para que dos aplicaciones puedan comunicarse), dentro de un link (para que se puedan comunicar dos dispositivos que comparten un mismo link), de ámbito local (para que dos dispositivos pertenecientes a una misma red puedan comunicarse), o globales (dentro de Internet).
- **Direcciones *Multicast*:** una misma dirección es asignada a más de una interfaz. Un paquete dirigido a este tipo de direcciones es entregado a cada interfaz a la que se le asignó esa dirección. Una dirección *multicast* no podrá ser el campo origen de un paquete, porque si no, si se espera respuesta se respondería a todas las interfaces con esa dirección. Esto puede ser utilizado como ataque a una red. IPv6 reserva varias direcciones de este tipo dependiendo del ámbito; así por ejemplo, se reserva una dirección *multicast* para poder enviar mensajes a todos los nodos que compartan un link, también se reserva otra dirección para que todos los *router*s reciban mensajes, y así poder descubrir prefijos.
- **Direcciones *Anycast*:** al igual que antes, una misma dirección es asignada a más de una interfaz, pero con este tipo de direcciones sólo será una interfaz la

que reciba el mensaje. La implementación es la que decide cuál de todas será la que lo reciba. Este tipo es útil en el caso de que el paquete pueda ser recibido y procesado por varios nodos, ya que proporciona redundancia.

- **Direcciones *no especificadas*:** es una dirección que no indica ninguna parte específica de la topología, ya que los 128 bits son cero.
- **Direcciones IPv6 que contienen direcciones IPv4:** IPv6 se va introduciendo gradualmente, por lo que actualmente conviven IPv6 e IPv4 y se necesita que ambos tipos de direcciones puedan ser usadas indiferentemente. Por lo general, en los 32 bits menos significantes de las direcciones IPv6, se introducen las direcciones IPv4.

Dentro de IPv6 existe un protocolo denominado *descubrimiento de vecinos*, que incluye desde la resolución de direcciones (ARP en IPv4) hasta encontrar los nodos que están en el mismo link, pasando por la autoconfiguración de la dirección IP de un nodo. Es necesario conocer tanto la dirección de los nodos que comparten tu mismo link para poder comunicarte con ellos, como la del *router* por defecto, para poder enviarle a él los paquetes dirigidos a nodos que no estén dentro de tu mismo link, y de este modo que él los pueda encaminar. Este protocolo permite descubrir los *router*s por defecto de una manera dinámica, ya que un nodo puede estar conectado a más de un *router*, y depende del destino deberá ser encaminado por uno o por otro.

3. 1. 2 MOBILE IPv6

Se puede definir la movilidad IP como el cambio de la dirección IP de un nodo debido a un cambio de su punto de unión dentro de la topología de Internet [2]. Estos cambios pueden ser debidos tanto por un movimiento físico del nodo, como por un cambio en la topología de la red, entre otros motivos. Si se produce alguno de estos cambios, se pueden originar varios problemas derivados de ellos: al estar manteniendo una comunicación con otro nodo, dicho nodo puede estar enviando los mensajes a la dirección IP antigua, y dichos paquetes se perderán, y con ellos la comunicación entre ambos dispositivos. También, el nodo que se desplaza necesitará generarse una nueva dirección IP para poder comunicarse con otros nodos, y por lo tanto se tendrá que crear un nuevo *socket*. Este tipo de problemas han ido surgiendo debido a la proliferación de los dispositivos *wireless*, o a las redes *peer-to-peer*, y evidentemente necesitan ser solucionados, que es precisamente lo que se pretende con Mobile IPv6. Dicho protocolo se desarrolla a nivel de red.

Por lo tanto, con Mobile IPv6 se pretende que al cambiar la ubicación de un nodo no se interrumpan las comunicaciones que se estén manteniendo, que un nodo siga siendo alcanzable por otros, ser independiente de la aplicación que esté usando la capa de red, ser independiente del nivel de enlace, y por supuesto, mantener el mismo rol que tuvieses antes

del cambio (*host*, o *router*). Para conseguirlo, siempre se mantiene una dirección IP fija, denominada *home address* a la que serán enviados todos los paquetes, y posteriormente, los paquetes destinados a dicha dirección serán redirigidos a la dirección actual del nodo.

Para conseguir el redireccionamiento, son necesarios los siguientes componentes:

- ***Home address:*** dirección permanente de los nodos.
- ***Home link:*** enlace en el que se encuentra la *home address*.
- ***Home Agent:*** *router* en el *home link* que actúa de intermediario entre el nodo móvil y cualquier otro nodo que quiera comunicarse con él.
- ***Foreign link:*** cualquier link (excepto *home link*) que visite el nodo móvil.
- ***Care-of address:*** dirección asignada al nodo móvil cuando se encuentra en cualquier *foreign link.*
- ***Binding:*** asociación entre el *home address* y el *care-of address*, realizada con el propósito de que el nodo móvil pueda ser alcanzado independientemente del link en el que se encuentre.
- ***Binding cache:*** estructura en el cual se almacenan todos los *bindings* de un nodo.
- ***Binding update list (BUL):*** lista de los *bindings* que fueron enviados al *home agent* y a los nodos que trataron de comunicarse con el nodo móvil a través de dicho *home agent*. Es mantenida por el nodo móvil.

En Mobile IPv6, el *home agent* es uno de los elementos fundamentales para que cualquier nodo de la red pueda establecer conexión con un nodo móvil. Él se encarga de que un nodo móvil sea alcanzable por los demás, manteniendo una entrada fija en el servidor DNS, y además, oculta la movilidad a las capas superiores al nivel de red. Si el *home agent* de un nodo falla, dicho nodo será inalcanzable por los demás nodos. Es evidente que, si un nodo cambia frecuentemente de dirección IP, puede que los diferentes servidores DNS tengan diferentes direcciones IP de un mismo nodo. Al mantener una dirección fija, cuando un nodo quiera comunicarse con el nodo móvil, siempre enviará los paquetes a la dirección del DNS, y por lo tanto no conocerá si el nodo móvil está ubicado en el *home link* o en algún *foreign link,* ya que los paquetes siempre son enviados al *home link.* Si el nodo móvil no se encuentra el *home link*, el *home agent* se encargará de realizar *tunneling* a los paquetes destinados a él, y posteriormente enviárselos. IPv6, únicamente será invocado cuando el nodo se encuentre fuera de su *home link.*

Para que los nodos puedan localizar a un *home agent* sus direcciones pueden estar almacenadas en la memoria no volátil del nodo, o bien se puede utilizar *Dynamic Home Agent Address Discovery* (*DHAAD*). Es aconsejable utilizar el segundo mecanismo, ya que las direcciones IP de los *home agent* pueden variar. En algunos casos una configuración manual seguirá siendo necesaria, ya que por ejemplo si la red donde se encuentra el *home agent* es renombrada el nodo móvil no será capaz de comunicarse con él.

Cuando un nodo cambia su ubicación, lo primero que deberá conseguir será una nueva dirección IP (para ser alcanzable en el nuevo link), que contenga el prefijo de dicho link. A continuación, el nodo móvil informa al *home agent* del cambio de posición, y por lo tanto de su nueva dirección, enviando un *binding Update* (*BU*). Este mensaje incluirá el *home address* del nodo (para poder identificarlo), y su *care-of address* (para poder localizarlo en su nueva ubicación). Cuando un *home agent* recibe un *BU*, éste tomará una serie de medidas para autenticar el mensaje. Una vez autenticado, el *BU* será aceptado, e incluido en la *BUL*. Un nodo móvil únicamente podrá enviar un *BU* a un único *home agent*, aunque podrá tener varias *home addresses* en varios *home agent*.

Al recibir el primer *BU* de un nodo, el *home agent* deberá asegurarse que la *home address* del nodo que envía el *BU* es única. Una vez que un nodo envía un *BU* a un *home agent,* tiene que esperar la confirmación de la recepción del mensaje. Este mensaje, denominado "*binding acknowledgment*" (BA), indica al nodo que envió el *BU,* que el *home agent* ha realizado correctamente la asociación entre su *home address* y su *care-of address*. Si el nodo no recibe este último mensaje, continúa mandando al *home agent* el mismo *BU*, en períodos de tiempo más espaciados, hasta que éste le confirme su recepción. En el momento que se recibe, almacena la nueva información en su *BUL*, de esta forma el nodo podrá saber cuándo una dirección está a punto de perder su validez, ya que su *lifetime* está a punto de expirar, y así poder informar al *home agent* que dicha dirección sigue siendo válida (en caso de ser necesario).

El *home agent* actúa como un *proxy*, representando al nodo móvil en el *home link*. Para ello, el *home agent* envía un mensaje a la dirección *multicast* del *home link*, indicando que todos los paquetes que se quieran enviar al nodo móvil, es decir a la *home address*, deberán ser enviados previamente a él (*proxy neighbor advertisement*). Además, el *home agent*, también defiende la *home address* del nodo móvil mientras él no se encuentra en el *home link*. En el caso en el que un nodo intente configurarse la *home address*, será el *home agent* el que responderá indicando que dicha dirección es la dirección de un nodo en el link.

Una vez que el *home agent* recibe un paquete destinado a la *home address* del nodo móvil, éste busca en su caché la *care-of address* de ese nodo, realiza el *tunneling* al paquete, y lo redirecciona a la nueva ubicación del nodo. El *túnel* es bidireccional, con lo cual el nodo móvil también tendrá que realizar *tunneling* del paquete que envíe al *home agent*, éste lo desencapsulará y lo enviará al destino.

Es importante, que el nodo móvil envíe los *BU* una vez que él detecte el movimiento, puesto que en el caso de retrasarlo, el *home agent* podría seguir enviándole los paquetes a la antigua posición, en la que ya no se encuentra, y por lo tanto esos paquetes se perderían. El nodo móvil se percata de que ha cambiado de posición cuando recibe un mensaje de un *router* nuevo, cuyo prefijo será diferente al anterior, o cuando el *router* por defecto haya desaparecido. No es necesario que el nodo se haya movido para que se produzca alguno de los dos casos anteriores, ya que en el primer caso un *router* puede usar dos prefijos distintos para el mismo link, y en el segundo puede que el nodo este unido a dos *router*s, y primero reciba

noticias de uno, y después de otro. Más aún, si cambias de ubicación pero sigues unida al mismo *router*, no implicaría ningún movimiento topológico.

Cuando un nodo móvil, regresa a su *home link,* deberá avisar al *home agent* que desde ese momento él será el que reciba los paquetes destinados a su *home address*. Para conseguirlo, el nodo móvil envía un *BU* al *home agent* indicando que su *care-of address* es igual a su *home address*, y con *lifetime* igual a cero. En este punto se plantea un conflicto, dado que el *home agent* tiene que "defender" la *home address* del nodo móvil y éste la quiere recuperar. Para solucionarlo, el nodo móvil envía un paquete al *home agent* con dirección origen la *home address* y dirección destino la dirección del *home agent*. Una vez recibido el *ack* del *home agent,* el nodo móvil enviará un mensaje *multicast* indicando que los paquetes con destino a su dirección IP, le sean enviados a él en lugar de al *home agent*.

Destacar, que cuando un nodo cambia de ubicación, no sabe si esa nueva ubicación se encuentra en la misma red local en la que estaba antes o si se encuentra en otra, por lo que no podrá utilizar las propiedades de la red, como puede ser el prefijo, a la que se encontraba unido para crearse su nueva *care-of address.*

Los mensajes pertenecientes a IPv6 (como por ejemplo el *binding update*) se encuentran codificados como opciones en una nueva cabecera, denominada "Cabecera de Movilidad". Entre los campos más importantes se encuentran: "*payload proto*", que indica el tipo de la siguiente cabecera, "longitud de la cabecera", "tipo de la cabecera de movilidad", que anuncia que tipo de mensaje está incluido en la cabecera, y "*checksum*".

3. 2 Proxy Mobile IPv6

Como se ha visto en la sección anterior, Mobile IPv6 propone una solución al problema de que un nodo pueda cambiar su ubicación dentro de una red. Para ello, es imprescindible que se mantenga una asociación entre la *care-of-address* de un nodo y su respectiva *home-address*, denominada *binding*. Además, se requiere que el nodo móvil envíe su nueva dirección, una vez se haya producido el cambio de ubicación a su *home- agent*, que será el que almacenará sus *bindings.* Por lo tanto será necesario que el nodo móvil sea consciente de todos los movimientos que realice, y además será el responsable de generarse una nueva dirección IP en su nueva ubicación [4].

Proxy Mobile IPv6 (PMIPv6) propone otro enfoque para resolver el mismo problema, extendiendo los mensajes de señalización entre un nodo y un *home-agent*, y que además no involucra al nodo móvil, es decir, no será él quien envíe los mensajes a su correspondiente *home-agent*. En este caso será un agente de movilidad intermedio quien envíe dichos mensajes, y gestione la movilidad en nombre del nodo móvil. De esta manera se obtiene una solución para el problema basada en la propia red. Recordemos que Mobile IPv6 la proponía a

nivel de nodo. Es aconsejable que todas las redes, tengan o no nodos móviles, fuesen capaces de soportar esta funcionalidad.

A lo largo de esta sección se explicará el funcionamiento básico de este nuevo protocolo. Para conocer más detalles sobre él, se remite al lector al anexo A de este documento.

Gracias a que PMIPv6 está basado en MIPv6 cuenta con dos grandes ventajas; poder reutilizar la funcionalidad que desempeña el *home-agent*, así como los mensajes que envía y su formato, y poder utilizar el *home-agent* como agente de movilidad para todos los tipos de nodos IPv6 existentes.

A lo largo del documento, cuando se nombre el dominio de PMIPv6 se referirá a toda aquella red, en la que exista movilidad y que el nodo móvil actúe como un nodo no móvil, dado que él puede que no sea consciente de su propio movimiento.

Los elementos principales de este protocolo, y que llevarán a cabo su realización son 2:

- **Local Mobility Achor** (LMA): tiene las mismas funcionalidades que tiene el *home-agent* de Mobile IPv6, y además soporta las nuevas funcionalidades aportadas por PMIPv6. Es la entidad que maneja el estado actual de los *bindings* del nodo móvil.
- **Mobile Access Gateway** (MAG): corresponde con una funcionalidad de un *router* de acceso, la cual se encargará de gestionar toda la señalización de movilidad de un nodo que esté conectado a uno de sus links de acceso. Por lo tanto, será el responsable de seguir la trayectoria del nodo móvil, así como de comunicárselo al LMA.

Todo dominio PMIPv6 deberá contar con estos dos elementos.

Además, para poder llevar a cabo la correcta ejecución del protocolo, serán necesarios definir los siguientes componentes:

- **Nodo móvil (MN):** *host* o *router* cuya movilidad es gestionada por la red. Puede ser IPv4, IPv6 o de las dos. Dado las especificaciones de este protocolo, no será necesario que el nodo móvil participe en la señalización de su movilidad.
- **LMA Address:** corresponde con la dirección del *Local Mobility Anchor*. Es a la dirección a la cual el *Mobile Access Gateway* enviará los mensajes de señalización oportunos sobre el movimiento de un nodo.
- **Proxy Care-of Address:** es la dirección configurada en la interfaz de salida del *Mobile Access Gateway*, y por lo tanto es el punto final del túnel establecido entre el LMA y el MAG. Para el LMA esta dirección es como la *care-of-address* de MIPv6, y por lo tanto tendrá que ser almacenada en la *Binding cache* (siendo ésta igual a la de Mobile IPv6) del LMA como entrada del nodo móvil.

- **Prefijo de *home network* del nodo móvil:** corresponde con el prefijo asignado al link entre el nodo móvil y el MAG. Puede que haya asignado más de un prefijo a dicho link, en cuyo caso los prefijos asignados serán tratados como un conjunto asociado con la sesión de movilidad. Por tanto el nodo móvil puede configurar su interfaz con uno o más prefijos del nombrado conjunto. Si el nodo móvil se conecta a través de varias interfaces simultáneamente a un dominio PMIPv6, a cada una de las interfaces se le asignará un conjunto único de estos prefijos y cada conjunto será gestionado por una sesión de movilidad diferente. Este tipo de prefijos pueden tener una longitud de hasta 128 bits.
- **Home Address del nodo móvil:** se denomina así a una dirección que pueda ser incluida dentro del prefijo de la *home network*. El nodo móvil podrá usar dicha dirección siempre y cuando se encuentre dentro de un mismo dominio PMIPv6. En caso de existir más de una dirección asignada a un mismo nodo, dicho nodo será accesible por cualquiera de sus direcciones. A diferencia de MIPv6, donde el *home agent* era consciente de la *home address* del nodo móvil, en PMIPv6, las entidades de movilidad únicamente son conscientes del/de los prefijo/s de la *home network* del nodo móvil, y en algunos casos de la dirección exacta asignada.
- **Home Link del nodo móvil:** es el link en el cual el nodo móvil obtiene su dirección IPv6 para la interfaz por la que se ha conectado una vez haya entrado en el dominio PMIPv6. Conceptualmente, es el link que siempre se mueve donde se mueva el nodo, por lo tanto, la red deberá asegurar que el nodo móvil siempre crea que él está dentro de ese link, y por lo tanto siempre tendrá esta misma dirección en cualquier link que esté ligado al mismo dominio PMIPv6.
- **Nodo móvil Multihomed:** se dice que aquel nodo que se conecta a un mismo dominio PMIPv6 a través de varias interfaces, usándolas simultáneamente.
- **Identificador del nodo móvil:** corresponde con el identificador del nodo móvil dentro de un dominio PMIPv6. Dicho identificador será permanente.
- **Identificador de la capa de enlace del nodo móvil:** representa la interfaz por la cual el nodo se conecta al dominio PMIPv6. Puede ser generado por el nodo y transmitido al MAG. Debe ser estable, es decir, invariable.
- **Perfil de política:** conjunto de parámetros de configuración para que un nodo móvil se comporte de una determinada manera.
- **Proxy Binding Update (PBU):** mensaje de petición enviado por el MAG al LMA del nodo móvil con el fin de establecer una asociación entre el/los prefijo/s de la *home network* del nodo móvil y su actual *Proxy care-of-address* (que correspondería con la *care-of-address* de Mobile IPv6).
- **Proxy Binding Acknowledgement (PBA):** mensaje de respuesta enviado por el LMA al MAG respondiendo a un PBU recibido.
- **Modelos:** existen dos tipos de modelos de direccionamiento que pueden ser usados:
    - **Per-MN-Prefix:** el/los prefijo/s asignados a cada nodo son únicos.

- **Shared-Prefix-Model**: los prefijos son compartidos por más de un nodo.

PMIPv6 soporta el modelo Per-MN-Prefix.

- **Sesión de Movilidad:** en el contexto de PMIPv6 éste término hace referencia a la creación o la existencia de un estado asociado con las asociaciones de movilidad de un nodo móvil en el LMA y en el MAG.

3. 2. 1 Funcionamiento de PMIPv6

Con PMIPv6 se pretende dar una solución al problema de movilidad de un nodo basado en la red, en la que el nodo no participe en la notificación de su movilidad, basándose en el protocolo Mobile IPv6. Las entidades de movilidad serán las encargadas de seguir la trayectoria del nodo móvil, de inicializar el intercambio de mensajes, además de crear el correspondiente estado del nodo, para así poder encaminar los mensajes destinados a él. Por lo que se pueden considerar que el LMA y el MAG son las entidades más importantes para el correcto desarrollo del protocolo PMIPv6.

Así, el LMA es el responsable de asegurar que el nodo móvil pueda ser siempre alcanzable por otros nodos, mientras que el MAG es la entidad que lleva a cabo la gestión de la movilidad de un nodo en su nombre. El MAG reside en el link de acceso en el cual el nodo está unido al dominio PMIPv6. Además, detecta el movimiento desde y hacia el enlace de acceso para poder así iniciar el intercambio de mensajes y dar de alta la nueva ubicación en la *binding cache* del LMA.

Dentro de un dominio PMIPv6 puede haber muchos LMA, pero cada uno de ellos sirve a grupos diferentes de nodos móviles.

Cuando un MN se une a un dominio PMIPv6, mediante un link de acceso, el MAG de ese link será el responsable de identificar al nodo y autorizar su movilidad. En el caso en el que pueda ser un MN, la propia red ha de asegurar que el nodo utilizará algún mecanismo de configuración (como puede ser DHCP) para así poder desplazarse a otra ubicación dentro de la red. Dicha configuración incluirá la/s dirección/es incluidas en el prefijo de la *home network* del nodo, el *router* por defecto al que enviará todos sus paquetes, y los parámetros necesarios de configuración. Aunque uno de los parámetros de configuración de la política a seguir por un nodo indica el tipo de dirección/es aceptadas, en este caso únicamente se aceptarán direcciones IPv6.

Si el nodo móvil se conecta a través de varias interfaces al dominio PMIPv6, la red asignará un conjunto único de prefijos de *home address* a cada interfaz, y el propio nodo será el que configure la/s dirección/es dentro del/de los respectivo/s prefijo/s de cada *home address*. Cuando un nodo cambia su configuración actual por otra configuración

correspondiente a otra interfaz por la que se conecta a la red, se produce un *handoff* (relevo). Esta operación puede producirse también si el nodo móvil cambia de MAG, al cambiar su punto de enlace con la red, aunque esté usando la misma interfaz. En caso de producirse un *handoff,* entonces, el MAG enviará un aviso al LMA, y éste último asignará al nodo los mismos prefijos de la *home network* anterior al relevo.

La siguiente figura muestra el flujo de mensajes intercambiados entre las entidades de movilidad y el MN cuando el nodo entra a formar parte del un dominio PMIPv6. Una vez que el nodo se ha unido a un link de acceso envía una petición para conocer su *router* por defecto, encontrando así un MAG, éste le enviará al LMA un PBU, para dar de alta el nuevo nodo. El LMA contestará con un PBA para confirmar la creación de una nueva entrada en la *binding cache* correspondiente con el nuevo nodo. Una vez recibidas dichas notificaciones, se crea el túnel bidireccional entre el LMA y el MAG. Establecida la tabla de enrutamiento para el nodo móvil, el MAG ya poseerá toda la información necesaria para poder simular el *home link* del nodo. Por último el MAG enviará un último mensaje al nodo móvil notificándole tanto los prefijos de su *home network* como los prefijos de su *home link*, para así proceder a la configuración.

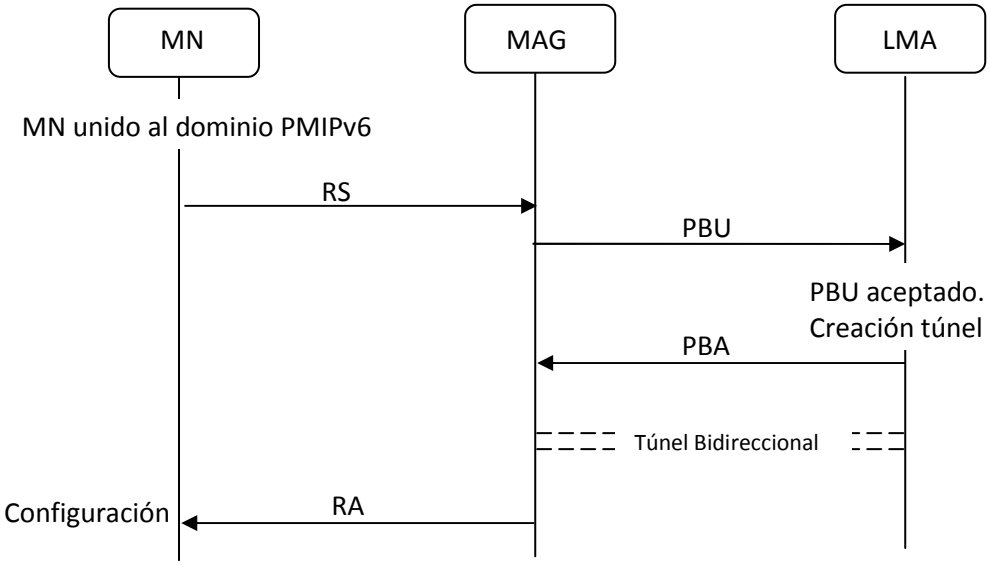


Figura 2: Señalización PMIPv6 al conectarse un MN

Una vez realizada la configuración del nodo, tanto el MAG como el LMA podrán manejar tanto los mensajes destinados al nodo móvil como los originados por él. Es el LMA la entidad que recibe todos los paquetes enviados al nodo móvil, él los redirigirá por el túnel hacia el MAG correspondiente. En este lado del túnel se borrarán las cabeceras ajenas al dominio en el que nos encontramos, y se enviará el paquete resultante al link de acceso del nodo móvil.

Dado que el MAG actúa como el *router* por defecto del MN, será a él a quien envíe los paquetes que desee. Éste los enviará al LMA por el túnel bidireccional creado entre ambos, quien eliminara las cabeceras externas al dominio y encaminará el paquete al nodo destino.

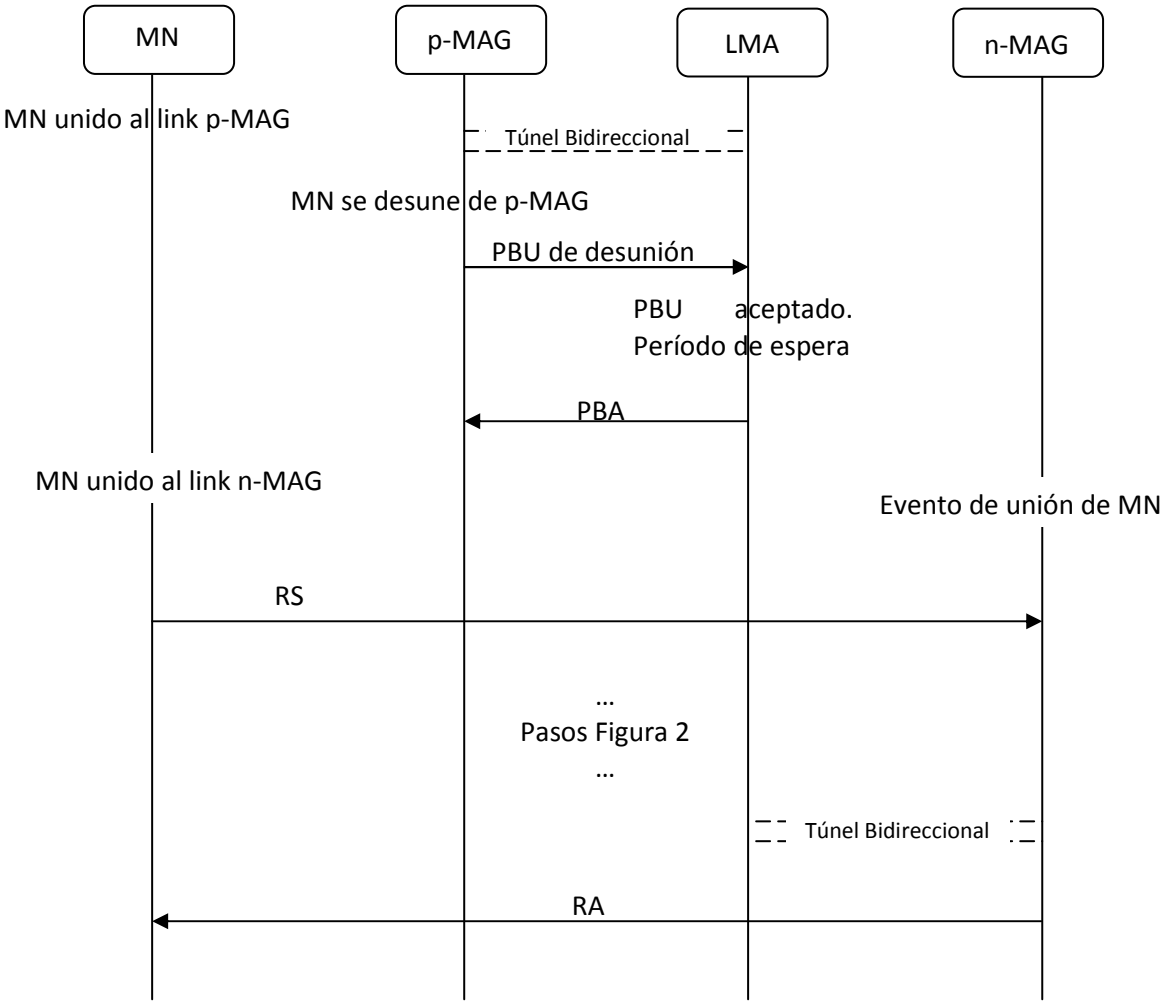


Figura 3: Señalización PMIPv6 cuando MN cambia de ubicación

La figura número 3 muestra los mensajes que se envían cuando se produce un *handoff* del nodo. Esto únicamente podrá suceder si el nodo está previamente registrado y configurado. El MAG inicial del nodo móvil es p-MAG, por lo que el nodo se encontrará en su link de acceso, cuando se mueve a otro link, p-MAG notará que el nodo ya no se encuentra en su link, por lo que eliminará el *binding* correspondiente con ese nodo y lo notificará al LMA a través del túnel. Al recibir la notificación, el LMA localizará la respectiva sesión de movilidad y esperará un determinado periodo de tiempo a que otro MAG, en este caso n-MAG le envié un PBU anunciando que ese nodo se ha unido a su link de acceso. En caso de no recibir ningún PBU en ese tiempo, el LMA borrará la entrada del nodo en la *binding cache.* Cuando un MAG detecte la presencia de un nuevo nodo en su link, es decir, cuando el nodo nuevo envíe una solicitud de *router*, el flujo de mensajes enviados será igual que en la figura anterior.

Posterior a ese intercambio de mensajes, la contestación del n-MAG a la solicitud del nuevo *router* será enviada al nodo, conteniendo los prefijos de la *home network* del nodo móvil, asegurando por tanto que el MN no percibe ningún cambio de link, y por consiguiente ningún cambio de ubicación, manteniendo así su configuración anterior.

Cabe destacar que el orden reflejado en la figura es el orden lógico de recepción de mensajes, pero no tiene por qué ser así siempre, ya que por ejemplo, puede suceder que el mensaje del n-MAG notificando que el MN se ha unido a su link, sea recibido por el LMA antes que el mensaje enviado por el p-MAG notificando que el nodo móvil ya no se encuentra en su link [5].

CAPÍTULO

4

TRABAJO REALIZADO

Nunca vayas por el camino trazado, porque conduce hacia donde otros han ido ya.
Alexandre Graham Bell (1847-1922)

4. 1 Desarrollo de PMIPv6

A lo largo de este apartado se tratará de explicar el motivo del porqué de la implementación de una versión reducida del protocolo PMIPv6, la funcionalidad que esta versión deberá llevar a cabo, así como el funcionamiento esperado del mismo.

La decisión de implementar una nueva versión de PMIPv6, surge tras los problemas encontrados durante los pasos dados con anterioridad. Según se va desarrollando el proyecto, se van encontrando más y más dificultades, que se fueron resolviendo, pero llegó un punto en el que se consideró más eficiente parar de resolver problemas y empezar de nuevo todo el desarrollo. Entre los problemas más importantes que se encontraron cabe destacar:

- El programa, inicialmente fue desarrollado por unos japoneses, con la intención de que implementase MIPv6. A continuación, la Universidad de Helsinki, tomando como base dicho código, lo amplió para que se ejecutase PMIPv6. Por último, se sabe que varias empresas continúan modificándolo para aumentar su funcionalidad. Con todo ello, se consigue que la legibilidad del código no sea toda la que debería ser, debido a que ha sido por diferentes personas de diferentes empresas cuya manera de trabajar, a su vez, es distinta.
- Además, no existe documentación de dicha implementación, con lo cual dificulta aún más su entendimiento, debido a la gran dimensión del mismo. Aunque, el código ha sido comentado, no se considera suficiente.
- Siempre que se encontraba algún problema de índole menor (como algunos mencionados en secciones anteriores), se iban corrigiendo aisladamente, es decir, sin tener en cuenta otras partes del programa, lo que en jerga se conoce como, “poner parches”.
- Pero el problema que desencadenó esta decisión fue sin lugar a duda la aparente aleatoriedad del funcionamiento del programa al introducir una nueva funcionalidad. Se observaba dicho comportamiento, tanto en el programa inicial como en el programa en el que se había añadido dicha funcionalidad, siempre que el programa instalado en los puntos de acceso estaba siendo ejecutado.

Todo ello, hace que la estabilidad de dicho programa no sea la esperada, debido a que en ciertas ocasiones se comportaba de manera incomprensible, ya que realizando las mismas operaciones que en otras situaciones, y con los mismos datos, no se obtenía el mismo resultado. Depurar este tipo de fallos es de una gran complejidad, ya que para ello se debe conocer muy bien el programa, además, se añade una dificultad mayor debido a lo que se ha comentado anteriormente, por lo que no resulta una tarea sencilla de realizar.

Si se consigue un programa estable, en el que siempre que se diesen ciertas circunstancias se pudiese asegurar que la comunicación entre el MN y el LMA se establece,

entonces se podrá examinar el comportamiento de PMIPv6 de una manera exhaustiva. Además, se considera necesaria que esta característica se dé en una implementación de un protocolo de este tipo, en el cual, las consecuencias de un fallo puede llegar a ser crítico dependiendo de lo que el usuario esté realizando en la red en el momento en el que se produce dicho fallo.

Tal y como se comenta en el anexo C, en el entorno de pruebas creado, existen dos puntos de acceso a la red, son *routers* wireless Linksys WRT54GL, pero ambos han sido configurados para que únicamente realicen la función de punto de acceso a una red.

4. 1. 1 Alcance del sistema

El objetivo de esta nueva implementación, y como se ha comentado en la sección anterior, consistirá en conseguir una gran estabilidad, la cual únicamente lleve a cabo determinadas características de PMIPv6, que permitan realizar la funcionalidad básica del mismo.

Como funcionalidad básica se podría decir que se ha considerado los pasos que se deben realizar necesariamente, tanto en el MAG como en el LMA, para que se le pueda ofrecer el servicio de movilidad a un MN. En esos pasos estarán incluidos tanto las operaciones internas que se debe realizar en ambas máquinas como el intercambio de mensajes entre ambos.

Debido a que, dependiendo del rol que desempeñe una máquina (LMA o MAG), deberá realizar diferentes operaciones, en este caso, serán también dos programas diferentes, uno para cada tipo de rol.

4. 1. 1 1MAG

La funcionalidad que deberán llevar al cabo los MAGs será:

- Leer un fichero de configuración para obtener cierta información necesaria para el desarrollo del protocolo como pueden ser los nodos móviles a los que se les proporciona movilidad, así como su LMA, y determinados datos de la máquina en la que se va a ejecutar el programa.
- Controlar los mensajes que el programa instalado en el *router* envía a los MAGs. Estos mensajes podrán ser tanto de conexión como de desconexión, por lo tanto será necesario reconocer a que tipo pertenecen así como el MN que ha producido el envío de dicho mensaje.
- En el caso que sea un mensaje de una nueva conexión de un MN a la red del MAG:
    - Identificar el MN, y el LMA que gestiona la movilidad de dicho MN.

- Enviar un PBU al LMA correspondiente con la información necesaria sobre el MN.
- Esperar el PBA de asentimiento del PBU. Una vez recibido, se deberá parsear para obtener ciertos datos de los que no se disponía del MN anteriormente.
- Dar de alta en la BUL.
- Añadir la ruta necesaria para poder comunicarse con el MN.
- Crear el túnel que comunique dicho MAG con el LMA correspondiente.
- Generar un RA con destino el MN para que éste último sea capaz de auto configurar la ruta hacia el MAG, y así al actuar como su *router* le enviará todos los mensajes a él, y éste a su vez al LMA.

- Si por el contrario, el mensaje recibido es un mensaje de desconexión de un MN, entonces:

  - Se identificarán tanto el MN como el LMA correspondiente, buscando la entrada del MN en la BUL, recuperando de esta forma todos los datos necesarios.
  - Construir y enviar un PBU al LMA con la información necesaria sobre el MN, y con el tiempo de vida de dicho PBU igual a cero.
  - Eliminar dicha entrada de la BUL.
  - Destruir túnel de comunicación con el LMA, siempre y cuando no sea usado por más MN.
  - Eliminar la ruta de comunicación con el MN.

4. 1. 1. 2 LMA

En el caso del LMA, se deberán realizar las siguientes operaciones:

- Mantener una lista de prefijos que puedan ser asignados a los diferentes MN cuando se reciba un PBU de un nuevo MN.
- Estar continuamente disponible para poder gestionar la posible recepción de un nuevo PBU en cualquier momento.
- Cuando se recibe un PBU con un tiempo de vida distinto de 0, entonces:

  - Se busca la entrada en la *Binding Cache*, para determinar si el MN que se ha conectado es un MN nuevo.
  - En caso de que no se encuentra una entrada en la *Binding Cache* de este nodo:

    - El MN se conecta por primera vez a nuestra red.
    - Se le asignará un prefijo para poder comunicarse con él.

- Se dará de alta la entrada en la *Binding Cache* con los datos obtenidos de parsear el PBU, añadiendo además el prefijo asignado.
- Se enviará un PBA de asentimiento de la gestión de la movilidad de este nuevo MN al MAG del cual recibimos el PBU.
- Se creará un túnel de comunicación con el MAG que envió el PBU.
- Se creará una ruta para poder comunicarse con el MN. Esta ruta será elegida siempre que un paquete vaya con destino al prefijo asignado al MN. Para ello, dicho paquete se encaminará a través del túnel recientemente creado.

o En el caso en el que se encuentre una entrada correspondiente a dicho MN en la *Binding Cache*, implicará que el nodo se ha cambiado de punto de acceso (*router*) a la red, por lo que:

- Será necesario actualizar la entrada de la *Binding Cache* con los nuevos datos que se obtengan del nuevo PBU. El único que se ha tenido en cuenta es el de cambio de MAG, y por lo tanto se modificará la *CoA* del MN.
- Se modificará el final del túnel de comunicación con el MAG del MN, ya que ahora será otra máquina diferente.
- No será necesario modificar la ruta de comunicación con el MN, dado que se seguirá encaminando a través del túnel de comunicación con el MAG.

- Cuando se recibe un PBU con tiempo de vida 0, significa que el MN se ha desconectado de la red de un MAG, pero no se sabe si se volverá a conectar a otro punto de acceso de la misma red, y con ello a otro MAG. Por lo tanto:

o Se esperará un tiempo prudencial, en el cual se considera que un MN que se vaya a unir a otro punto de acceso de nuestra red, pueda ser capaz de hacerlo.
o Vencido ese tiempo:

- Si en ese tiempo se ha recibido un PBU haciendo referencia al mismo MN, y con tiempo de vida distinto de 0, se actualizará tanto la *Binding Cache* como el túnel, de la misma manera que se acaba de explicar.
- En el caso de que no se reciba ningún PBU haciendo referencia a ese MN, en ese tiempo, se asumirá que el MN se ha desconectado de la red para no volverse a conectar en un período de tiempo pequeño, y por lo tanto se procederá a eliminar tanto la entrada de la *Binding Cache*, como el túnel de comunicación con el MAG y la ruta para encaminar paquetes hacia el MN.

A diferencia de la implementación anterior, cabe destacar, que en este caso el LMA mantiene una lista de prefijos (en lugar del MAG), que irá asignando según vaya recibiendo PBUs de nuevos MN. Anteriormente esto era realizado mediante el fichero de configuración del MAG, ya que era necesario introducir en él, los nodos móviles autorizados a utilizar PMIPv6 y por cada uno de ellos anunciar también el prefijo que se le iba a asignar. Por lo tanto al iniciar el programa y leer el fichero de configuración, a todos los nodos móviles se les creaba un perfil, y en el momento que se conectaba a la red del MAG, ya se sabía que prefijo le iba a ser adjudicado.

Asimismo, en esta nueva implementación, para que un nodo esté autorizado a utilizar PMIPv6 bastará con que se encuentre en el fichero de configuración. Al igual que anteriormente, se les creará un perfil a cada uno de ellos con la información necesaria para poder prestarles el servicio, y cuando un nodo se conecta a la red, únicamente se mirará si se tiene dicha información de él, y en caso de no ser así se ignorará. En la implementación anterior, además de eso se comprobaba si el nodo estaba autorizado a PMIPv6.

A la hora de dar de baja un nodo móvil, porque ya no pertenezca a la red de alguno de los MAGs, se utilizará el programa que ha sido instalado en los puntos de acceso de la red, tal y cómo se explica en el Anexo E. Tanto cuando se conecta un nuevo nodo a la red como cuando se desconecta de ella, el punto de acceso enviará un mensaje comunicándoselo al MAG al cual está conectado. Por lo tanto, si un MAG recibe un mensaje de desconexión de un MN, se lo comunicará al correspondiente LMA enviándole un PBU con tiempo de vida 0. Es entonces cuando el MAG eliminará la entrada de dicho nodo de la BUL, y el LMA esperará un tiempo prudencial para comprobar que efectivamente ese MN se ha desconectado de la red, y no se ha cambiado de punto de acceso. De ser así, procederá a eliminar su entrada de la *Binding Cache*. A continuación ambos eliminarán las rutas y túnel, relativo a dicho MN.

Tanto en la *Binding Cache* como en la BUL únicamente se almacenarán los datos que han sido considerados necesarios para llevar a cabo esta versión reducida del protocolo. Estos datos son muchos menos de los que se guardaban para llevar a cabo la implementación original, dado que también la funcionalidad es más reducida. Dichos datos serán detallados a continuación, en la sección 4.4 de este mismo capítulo.

4. 2 Funcionamiento

Para abordar esta sección, se representara un diagrama por cada uno de los roles que se pueden dar en PMIPv6 (excepto MN, ya que en este protocolo no es necesario que él disponga de un software específico) que muestran de una manera más visual el funcionamiento de ambos. Además, se considera que de esta forma el entendimiento del mismo también es mejor.

En ellos se mostrarán las diferentes acciones que va realizando cada nodo, dependiendo de lo sucedido con anterioridad desde el inicio del mismo.

Ambos diagramas representan una ejecución normal del programa, es decir siempre y cuando no sea interrumpido. Por eso mismo, no se ha añadido la finalización del mismo, dado que estará permanentemente ejecutándose, hasta que se decida parar. Para detenerlo, será necesario hacerlo mediante el comando de Linux CRTL+C. Éste podrá ser invocado en cualquier punto de la ejecución, terminándola de inmediato.

4. 2. 2. MAG

Para iniciar el programa, al igual que otro programa normal, se deberá ejecutar en la línea de comandos, del ordenador que vaya a actuar de MAG y dentro del directorio dónde se encuentre el ejecutable (habiendo compilado el programa previamente), el comando:

#./pmipv6

Se ha considerado que antes de iniciar el programa en el MAG, no se encuentra conectado ningún MN a su red.

Como se puede observar en la *fig.* 5, lo primero que debe ser realizado por un MAG ha de ser la obtención de los datos necesarios para el correcto funcionamiento. Estos datos se encuentran en un fichero de configuración denominado "pmip.conf", por lo que será imprescindible leer dicho fichero y almacenar dicha información en las estructuras de datos oportunas, descritas más adelante.

A continuación, se deberá establecer la comunicación con el punto de acceso al que se encuentre conectado. Para ello se abrirá un *socket* UDP (dado que en el programa que se ejecuta en dicho punto de acceso el *socket* también es UDP), con un puerto determinado, y con la dirección de acceso local asociada a la interfaz por la que se encuentra conectado a dicho *router*. Este *socket* se abrirá en modo pasivo, es decir, estará permanente atento a si se reciben cambios en él, es decir si recibe algún mensaje del punto de acceso a través de él. Por dicho *socket* únicamente se recibirán los mensajes que envía el programa que se instaló en el punto de acceso.

Una vez creado dicho *socket*, en el momento que se reciba algo por él, se procederá a leer dicho mensaje, y a identificar de qué tipo es el evento recibido. Los eventos que se reciben pueden ser de dos tipos:

- *Tipo 1*: evento de conexión. Un nuevo nodo se ha agregado a la red del MAG.
- *Tipo 0*: evento de desconexión. Un nodo que se encontraba en la red del MAG la ha abandonado.

En ambos casos, y como información adicional, del mensaje se extraerá la dirección HW del nodo que se ha conectado/desconectado de la red.

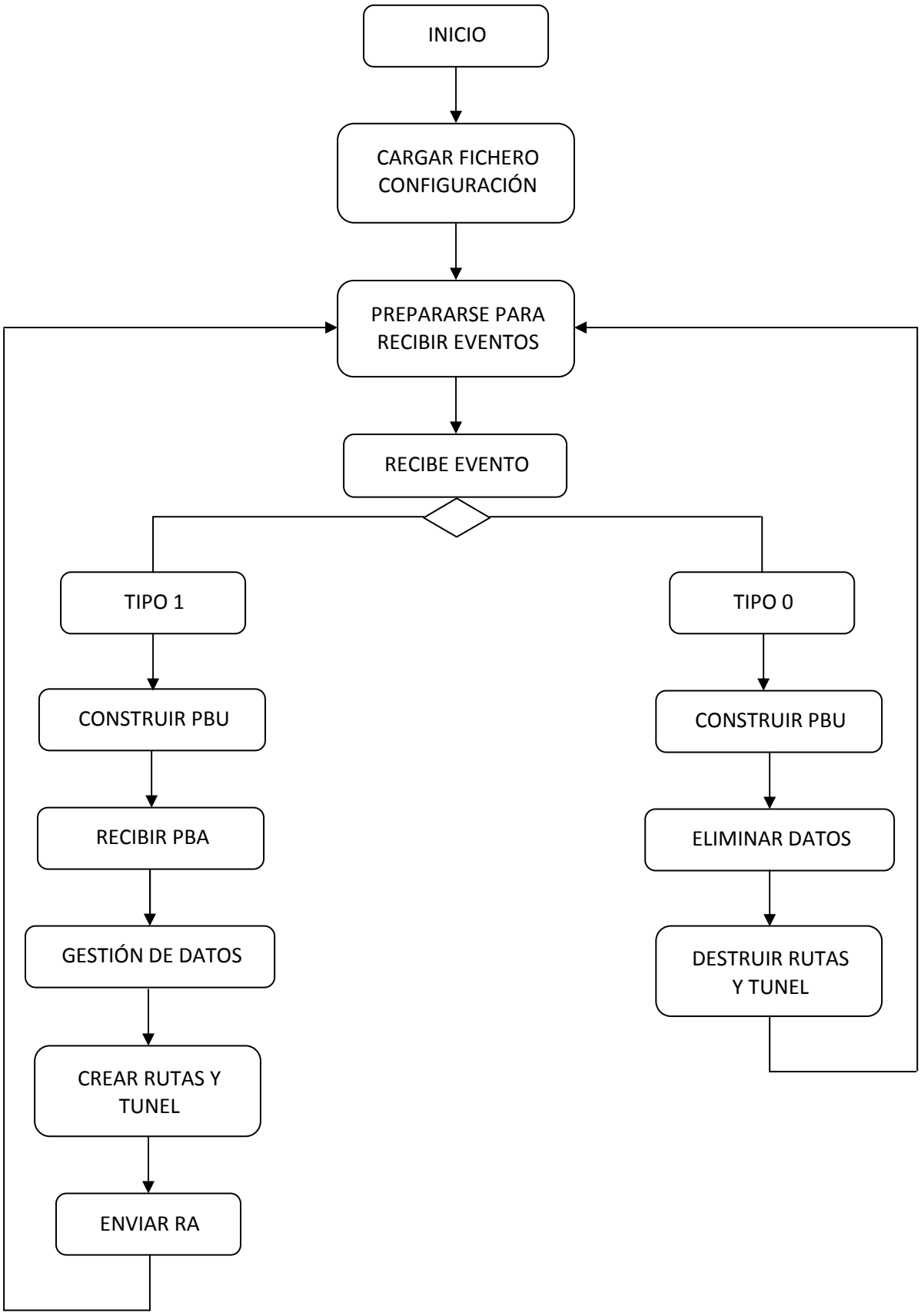


Figura 4: Diagrama actividad MAG

En el caso que el evento recibido sea de conexión, y se disponga de un perfil de dicho nodo, es decir se encuentre en el fichero de configuración, lo siguiente que se realizará será informar al LMA que se ha conectado un MN a la red, para que así éste pueda gestionarle su movilidad. No se comprobará si el MN se encontraba ya en la BUL, ya que se asume que no, puesto que en el momento que se desconecta se elimina todo lo relativo a dicho nodo. Siempre que se desconecta del punto de acceso al que se encontraba conectado se envía un evento tipo 0 al MAG. Para ello, se deberá construir un PBU con tiempo de vida distinto de 0. Para la construcción del PBU, y como se verá más adelante, se utilizará la estructura ip6_mh que representa la cabecera de movilidad para este tipo de mensajes. Para poder enviar estos datos, se necesitará un tipo de *socket* denominado RAW, el cual permite enviar cualquier tipo de mensaje a través de él. En este caso, todo lo que se escriba en el *socket*, se escribirá a continuación de la cabecera IP, que el *kernel* del ordenador incluirá, siempre y cuando no se haya incluido previamente.

En el momento que el PBU haya sido enviado correctamente al LMA, se esperará la respuesta de este último en forma de PBA. Con este tipo de mensajes el LMA acepta gestionar la movilidad del MN que se haya conectado. De él se pueden extraer datos necesarios sobre el MN, y que sin ellos no se podría llevar a cabo la ejecución de este protocolo, como es el prefijo que le ha sido asignado al MN, necesario para poder encaminar los mensajes destinados al MN.

Realizado todo lo anterior, el MN es dado de alta en la BUL del MAG correspondiente con todos los datos recopilados e imprescindibles para comunicarse tanto con el MN como con el LMA. Como ya se ha explicado, estos datos son obtenidos tanto del fichero de configuración como del PBA recibido.

A continuación se procede a crear la ruta hacia el nodo móvil. Esta ruta se crea mediante llamadas al sistema, para que sea éste el que ejecute el comando que se desea, diciendo que todos los paquetes con destino, una dirección que empiece por el prefijo asignado al MN, se mandarán directamente por la interfaz a la que el MAG se encuentra conectado al punto de acceso al que se conectó el MN. Para la creación del túnel se procederá de modo similar, es decir realizando llamadas al sistema, que en este caso son cuatro. Una para crear propiamente el túnel, otra para indicar que ese dispositivo puede ser utilizado, la siguiente para asignarle una dirección, que coincidirá con la dirección de acceso global de la interfaz por la que se está creando (es decir, la que se encuentra conectada al LMA), y por último para especificar que los paquetes con destino al LMA sean enviados a través del túnel que se acaba de crear, es decir, añadir una nueva ruta.

Por último, se deberá indicar al MN que el MAG es su *router* por defecto, para que todos los paquetes que envíe pasen por el MAG, y éste se los envíe al LMA, y ya sea este último el que los envíe a su destino. Esto se consigue enviando un RA a la dirección IPv6 de acceso local del MN. Para poder enviar este paquete, se ha de advertir al sistema que el MAG está directamente conectado al MN, es decir, es su "vecino". Es necesario, porque si no, de otro modo, y a no ser que el MN envíe un RS, el MAG no sabrá enviar el RA, y por lo tanto no

se enviará, y el MN enviará los paquetes que desee a otro nodo en lugar de al MAG, y por tanto, no se estará cumpliendo con el protocolo. Al igual que anteriormente, esto se realiza mediante una llamada al sistema. En el RA, además de anunciar al MN que el MAG es su *router*, se le advertirá del prefijo que ha de usar. Cuando el MN recibe este RA, configurará una dirección IPv6 que comienza por el prefijo que se le ha sido asignado.

Si por el contrario, el evento que se recibe es de desconexión, es decir, de tipo 0, primeramente se comprobará si el MN que ha provocado dicho evento tiene una entrada en la BUL del MAG que lo recibe. En caso afirmativo, se procederá a crear un PBU para informar al LMA que el nodo ya no se encuentra en la red, y que por lo tanto ya no se debe realizar los pasos propios de PMIPv6. Para ello, el tiempo de vida del PBU deberá ser 0.

Una vez enviado, no se precisará confirmación por parte del LMA, así que se procederá directamente a borrar la entrada de dicho MN de la BUL. Remarcar, que únicamente se borrará de la BUL, pero se seguirá teniendo el perfil de dicho nodo por si se vuelve a conectar a la red poder asegurar que se tienen los datos necesarios para empezar de nuevo todo el proceso.

Eliminada la entrada, se eliminarán las rutas que se habían creado para poder establecer comunicación con el MN y con el LMA. Además se eliminará el túnel que unía al MAG con dicho LMA. Ambas cosas serán realizadas de igual manera a como se crearon, es decir con las llamadas al sistema oportunas.

4. 2. 2 LMA

En el caso del LMA, el programa se inicia de igual manera que en el MAG, dado que el programa se llama igual.

En este caso, los datos que se necesitarán para que la labor del LMA pueda ser desarrollada correctamente es una lista de prefijos que se encontrará almacenada en él, y que será cargada una vez iniciado el programa. La estructura en la que se almacenará cada uno de ellos constará de dos campos, una para el propio prefijo, y otro campo que indicará si dicho prefijo ha sido asignado a un MN, en cuyo caso no se podrá volver a asignar.

Para poder recibir PBUs de los diferentes MAGs será necesario crear un *socket* RAW, y estableciéndole las opciones necesarias para que capture todos los paquetes que el ordenador reciba con dichas características. Para más detalles sobre este *socket* consultar el apartado “comunicación entre los nodos” de este mismo capítulo. Este *socket* permanecerá a la escucha permanentemente.

Cuando se recibe un paquete por el *socket* que se acaba de crear, siempre será un PBU. Por tanto, se sabe la estructura del mismo, por lo que se procede a extraer los datos de

mismo. Entre esos datos se encontrará el identificador del MN, por el cual se realiza una búsqueda en la *Binding Cache* del LMA para saber si el MN al que hace referencia dicho PBU ya se había dado de alta con anterioridad. Es importante conocer esto dado que el comportamiento del programa variará dependiendo del caso en el que nos encontremos.

El caso más sencillo será si el MN no está en la *Binding Cache*, lo cual indicará que dicho nodo es la primera vez que se ha conectado a alguno de los MAG asociados a la su red, y por tanto será necesario realizar todas las acciones para garantizar la conectividad del mismo. Entre ellas, las más importantes son las mostradas en la *fig.* 6.

La primera a llevar a cabo será asignar un prefijo disponible al nuevo MN. Para ello, se buscará en la lista de prefijos el primero que esté marcado como disponible. Una vez asignado el prefijo, y gracias a los datos facilitados por el MAG, como puede ser la CoA del MN, se habrá conseguido toda la información requerida. Es entonces cuando se procede a dar de alta dicho nodo en la *Binding Cache*.

Destacar que en esta implementación de una versión reducida de PMIPv6 se ha considerado que a cada MN se le será adjudicado únicamente un prefijo.

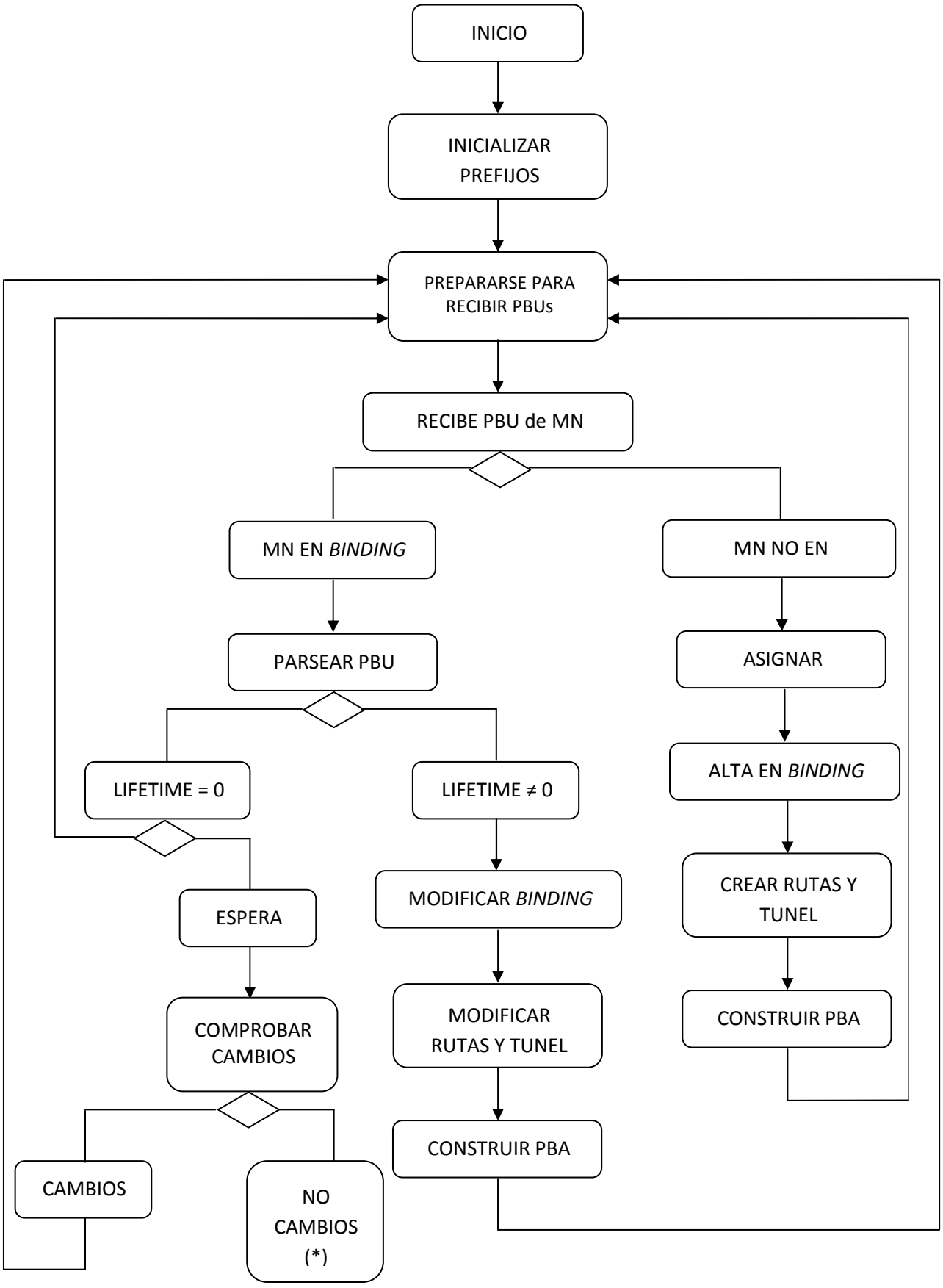
A continuación, lo primero que se realizará será crear el túnel que unirá al LMA con el MAG. Al igual que se realizaba en el MAG, esto es realizado mediante llamadas al sistema, para que se ejecute el comando que se desea. La información necesaria será únicamente la dirección del LMA, especificada como dirección local, la dirección remota, es decir el punto final del túnel, que coincidirá con la CoA, y la interfaz a la que se le asocia dicho túnel.

En este caso, únicamente será necesario crear una ruta, para indicar que todos los paquetes con destino una dirección IPv6 que comience por el prefijo asignado al MN que se acaba de conectar a la red, sean enviados a través del túnel que se acaba de crear.

Para finalizar el tratamiento de este PBU, será necesario indicar al MAG que se ha aceptado la petición del MN para gestionar su movilidad. Esto se realiza enviándole un mensaje PBA. La estructura de este mensaje es la misma que la del PBU, únicamente cambiando el tipo de mensaje que es, para indicar que corresponde con un PBA. Sólo se le añadirá una única opción, que coincidirá con el prefijo que se le ha sido asignado al MN. Es necesario especificarlo ya que, de no hacerlo, el MAG no podrá localizar al MN, y por tanto no sabrá como redirigirle los mensajes.

Si por el contrario, el MN ya se encontraba dado de alta en la *Binding Cache*, se puede recibir un PBU de un MAG por dos motivos:

1- Indicando que el MN se ha cambiado de punto de acceso a la red.
2- El MN se ha desconectado de la red.


INICIO
INICIALIZAR PREFIJOS
PREPARARSE PARA RECIBIR PBUs
RECIBE PBU de MN
MN EN *BINDING*
MN NO EN
PARSEAR PBU
ASIGNAR
LIFETIME = 0
LIFETIME ≠ 0
ALTA EN *BINDING*
ESPERA
MODIFICAR *BINDING*
CREAR RUTAS Y TUNEL
COMPROBAR CAMBIOS
MODIFICAR RUTAS Y TUNEL
CONSTRUIR PBA
CONSTRUIR PBA
CAMBIOS
NO CAMBIOS (*)

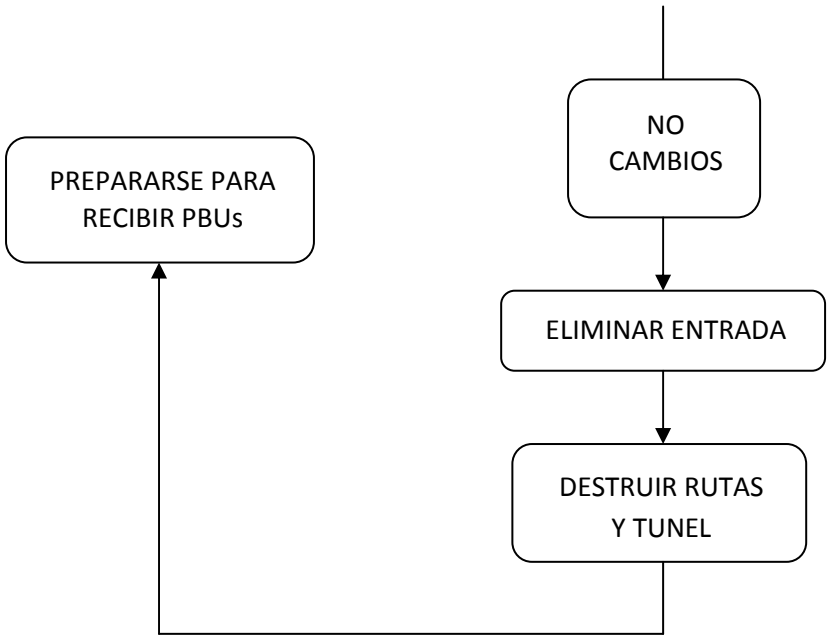


Figura 5: Diagrama actividad LMA

Para poder diferenciar cada caso, será necesario comprobar el tiempo de vida del PBU. En el caso que sea 0, el MAG estará indicando al LMA que un MN que antes estaba conectado a su red, se ha desconectado, y que por tanto él borrará tanto toda su información como las rutas y túnel que se habían creado para gestionar su servicio. En el caso que el tiempo de vida sea diferente de 0, indicará que el MN ha abandonado su anterior punto de acceso, y se ha conectado a otro perteneciente a la misma red.

Dado que el programa que se ha instalado en el punto de acceso enviará al MAG correspondiente un aviso de que un nodo se ha desconectado de la red, el MAG enviará siempre, en este caso un PBU al LMA con tiempo de vida 0. Si ese MN se conecta a otro punto de acceso de nuestra red, el nuevo MAG, enviará al mismo LMA un PBU con tiempo de vida distinto de 0 para indicar que un nuevo MN se ha conectado a su red. Por tanto, siempre que un MN cambie de punto de acceso el LMA recibirá dos PBUs uno con tiempo de vida 0, correspondiente al MAG antiguo del MN, y uno con tiempo de vida distinto de 0, correspondiente con el nuevo MAG del MN. Esto ha sido controlado para que siempre se gestione primero el PBU con tiempo de vida 0.

En el primer caso expuesto, es decir en el caso que se reciba un PBU con tiempo de vida distinto de 0, de un MN que ya se encuentra dado de alta en la *Binding Cache*, se actuará de una forma similar a cuando el MN no está en la *Binding Cache*. La diferencia es que en vez de crear lo necesario para gestionar su movilidad, se modificará. Por tanto, lo primero que se debe realizar es encontrar la entrada del MN en la *Binding Cache*. Una vez encontrada, será necesario modificarla, dado que la CoA será diferente. Además de la CoA será necesario modificar el campo de última modificación, para indicar que ha sido actualizada. Este campo es especialmente importante, dado que cuando se recibe un PBU con tiempo de vida igual a 0 se espera un tiempo prudencial para comprobar si el nodo se ha desconectado definitivamente

de la red, o si por el contrario únicamente ha cambiado de punto de acceso. Para poder saber esto, como se verá a continuación, se comprobará si el campo de última modificación ha sido modificado.

Al igual que cuando el PBU recibido corresponde con un MN que se conecta por primera vez a la red, el LMA deberá indicar al MAG adecuado que se acepta gestionar la movilidad del MN, e indicarle el prefijo que le corresponde mediante un PBA. El PBA es creado y enviado de la misma manera de la que se realizaba anteriormente. A diferencia con el caso anterior, no se le asigna un nuevo prefijo, se utiliza el que ya se le había asignado anteriormente, el cual se sabe dado que es una información del MN almacenada en la *Binding Cache*.

Puntualizar, que no es necesario modificar la ruta que se dirige al MN, ya que los mensajes destinados a él serán encaminados por el mismo túnel, y será este el que modifique su punto final al nuevo MAG.

De corresponder con el segundo caso, es decir el tiempo de vida del PBU es igual a 0, se lanzará un *thread* que realizará determinadas acciones para concretar si el MN ha abandonado definitivamente la red, y por tanto eliminar la información correspondiente a él, o si simplemente se ha movido a otro punto de acceso. El hilo principal del programa continuará a la espera de nuevos PBUs, dado que si el MN cambia de MAG, le será notificado al LMA mediante un nuevo PBU, aunque en este caso con tiempo de vida distinto de 0.

El nuevo *thread*, primeramente obtendrá toda la información almacenada en la *Binding Cache* relativa al MN que lo ha invocado. A continuación permanecerá dormido un tiempo prudencial, en el cual dicho MN pueda conectarse a otro punto de acceso de la misma red, para así seguir gestionando su movilidad. Este tiempo ha sido establecido inicialmente a 15s, pero podría ser cualquier otro, siempre que le diese tiempo suficiente para conectarse a otro MAG, que éste envíe el PBU al LMA, y éste le respondiese con un PBA.

Una vez transcurrido dicho tiempo, el *thread* volverá a recuperar la información del mismo MN de la *Binding Cache*. A continuación comparará si el campo de última modificación tiene el mismo valor que cuando lo obtuvo antes de la espera. Si dicho campo tiene el mismo valor, será porque no se ha recibido ningún PBU relativo a dicho nodo con un tiempo de vida distinto de 0, y por lo tanto se considerará que ha abandonado la red. En el caso de que los valores sean distintos, se asumirá que el MN sigue en la red ya que se habrá recibido un PBU de las características nombradas. Puede darse el caso de que el MN se desconecte de un MAG y se vuelva a conectar al mismo MAG, en cuyo caso el campo de última modificación también será actualizado. Por este motivo no se comprueba la CoA para decidir si se le sigue prestando los servicios de PMIPv6 a dicho MN.

Cuando el campo relativo a la última modificación de un MN ha sido actualizado en el tiempo en el que el *thread* está durmiendo, al despertar, éste no realizará ninguna acción, ya que el hilo principal del programa al recibir el PBU de actualización habrá realizado todas las

acciones necesarias para continuar gestionando la movilidad del MN, y por tanto directamente se continuará esperando nuevos PBUs.

Si por el contrario, los valores a comparar son los mismos, como ya se ha comentado, se considerará que el MN ha abandonado definitivamente la red, y por tanto no será necesario seguir manteniendo almacenada su información, ni la ruta para encaminar los paquetes dirigidos a él, y en caso de no ser utilizado el túnel por ningún otro MN, tampoco dicho túnel. Por lo que se procederá a eliminar todo. Una vez realizado esto, se continuará esperando nuevos PBUs de nuevos MN, o del mismo MN, aunque será tratado como si fuese la primera vez que se conecta a la red.

En el caso que se reciba un PBU con tiempo de vida 0 que no corresponda a ningún MN almacenado en la *Binding Cache*, simplemente se ignorará, dado que no se puede llevar a cabo ninguna acción de las comentadas.

4. 3 Comunicación entre nodos

Para que el protocolo funcione según lo esperado, es necesario que los diferentes equipos intercambien mensajes con determinada información necesaria para que la comunicación entre todos ellos quede establecida. El formato de estos mensajes está predeterminado, ya que son mensajes pertenecientes al protocolo IPv6, y deben ser reconocidos por cualquier nodo que soporte dicho protocolo.

A continuación se muestra un esquema con los diferentes mensajes que son enviados entre los nodos pertenecientes a nuestro entorno de pruebas.

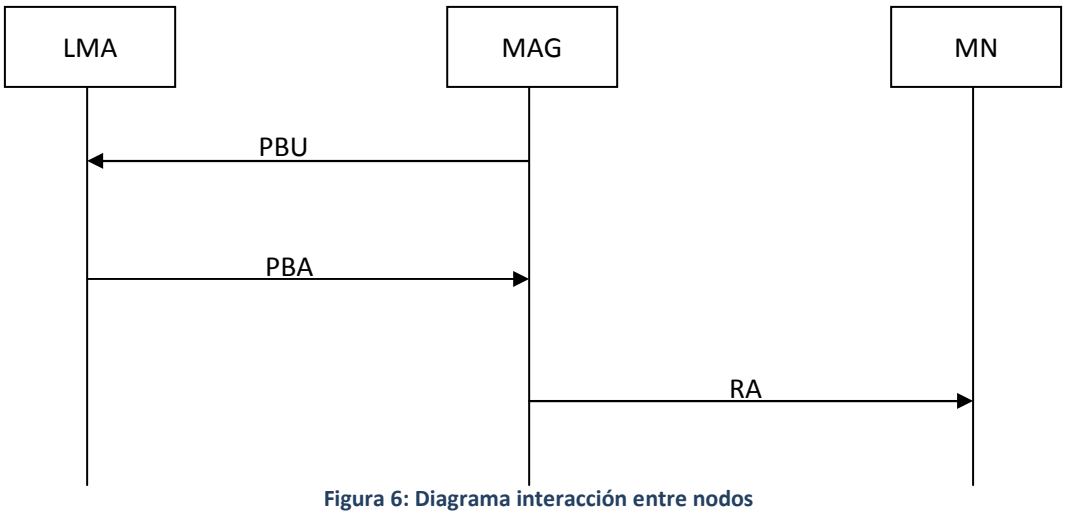


Figura 6: Diagrama interacción entre nodos

En este esquema se omiten los mensajes que son intercambiados entre los puntos de acceso, y los MAG, dado que no son generados por la nueva implementación de PMIPv6.

Los mensajes PBA y PBU, son mensajes de movilidad pertenecientes a IPv6, mientras que el mensaje RA es un tipo de mensaje de los denominados de control, incluido en el protocolo ICMPv6.

En todos los mensajes de movilidad, ha de ser incluida la cabecera de movilidad. Esta cabecera es identificada mediante el número 135 en el campo *next header* de la cabecera IPv6. Esto no afecta directamente a la programación, ya que al usar *sockets* de tipo *RAW*, todo lo que se escriba en él, será colocado a continuación de la cabecera IPv6, y el *kernel* de cada equipo será el responsable de construir correctamente tanto la cabecera IPv6 como de rellenar algunos campos de dicha cabecera de movilidad.

Tal y como se representa en [4], el formato de la cabecera de movilidad, será el siguiente:

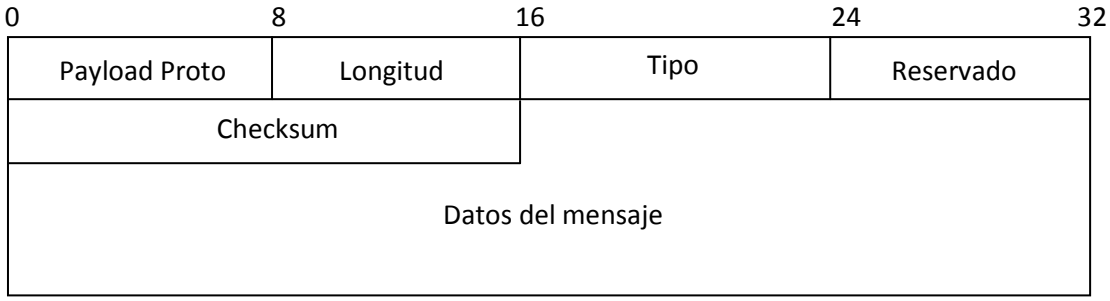


Figura 7: Cabecera de movilidad IPv6

Entre los campos que se encuentran en esta cabecera, en la implementación únicamente se rellenarán los necesarios, es decir, la longitud, el tipo, y los datos del mensaje que dependerán del tipo. El resto de los campos, es obligatorio que sean completados correctamente, como el *checksum,* dado que de no serlo, no podrá ser leído por el ordenador destino. En este caso, y en el de los demás campos, como se ha comentado anteriormente, es el *kernel* del equipo el que se encarga de realizarlo, siempre que hayan sido completados con el valor cero.

En el caso del campo *tipo*, podrá ser completado con dos valores, dado que únicamente se enviarán en esta implementación mensajes del tipo PBU o del tipo PBA. Cada uno de estos mensajes han sido previamente codificados, por lo que les corresponderá un valor numérico, que será con el que se deberá completar este campo, en vez de escribir directamente el número, se podrá utilizar una constante que haya sido declarada con ese valor. Estas constantes se encuentran definidas en la librería "*ip6mh.h*" que proporciona el lenguaje C. Así:

- PBU: IP6_MH_TYPE_BU= 5.
- PBA: IP6_MH_TYPE_BA= 6.

Dependiendo del valor con el que el tipo haya sido completado, los datos del mensaje variarán, dado que no se envía exactamente la misma información en un PBU que en un PBA, aunque algún dato pueda estar presente en ambos.

PBU

En el caso de que dicho campo haya sido completado con el valor 5, y de acuerdo con [4][5], un mensaje PBU corresponderá con el formato que muestra la siguiente figura:

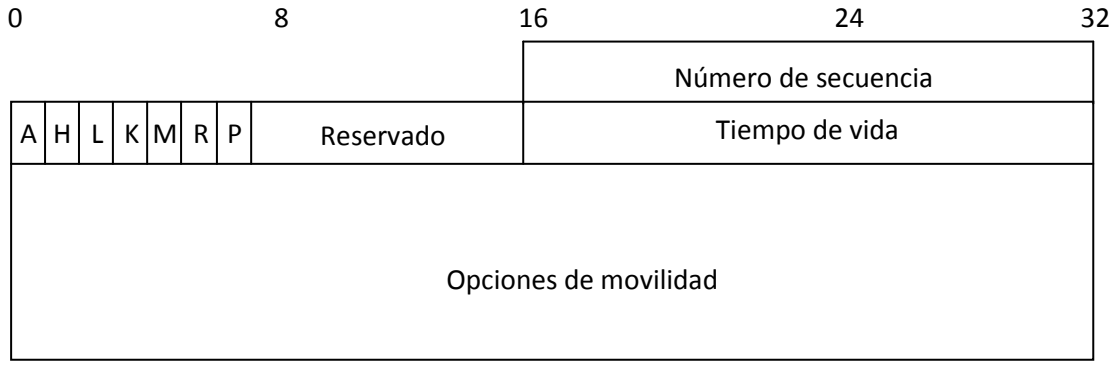


Figura 8: Formato PBU

El número de secuencia será relleno con un valor aleatorio, dado que en esta versión reducida de la funcionalidad de PMIPv6 no se tiene en cuenta su valor para realizar ciertas comprobaciones, como pudiera ser que PBU es más reciente. Recordar, que eso se podría realizar mediante el número de secuencia, o bien mediante la inclusión de un sello temporal en cada mensaje PBU; no se realiza ninguna de las dos en esta implementación. En este caso, y como ya se ha explicado con anterioridad, el encargado de realizar dichas comprobaciones será el LMA mediante un campo adicional en la *Binding Cache* que determinará la última vez que dicha entrada fue actualizada. Por tanto, el campo "número de secuencia" será completado para enviar el PBU, pero el valor con el que se rellene será indiferente.

Lo que diferencia a un mensaje PBU de un mensaje BU, es que al primero se le añade unos *flags* adicionales, ente los que destaca el *flag* P, que indicará que es PMIPv6 en vez de MIPv6. El *bit* A, indicará que una vez que el MAG envíe el PBU esperará recibir un PBA para completar el registro de un MN. El *flag* denominado con la letra H, será activado para indicar al destinatario del mensaje (es decir el LMA) que deberá actuar como HA para el MN [4]. En el caso de PMIPv6, será con el rol de LMA. Los *bits* especificados serán los *bits* que serán activados en esta implementación a la hora de enviar un PBU, los demás serán establecidos a 0 debido a que no se han considerado necesarios.

Al campo "Reservado" se le será asignado el valor 0, dado que actualmente está en desuso ya que pretende ser utilizado en futuras necesidades. El destinatario ignorará este campo.

Será el campo “tiempo de vida” por el que se distinguirán los dos tipos de PBUs que pueden ser enviados por esta aplicación. En el caso que el valor de este campo sea distinto de 0, el PBU indicará el registro de un nuevo MN. En caso contrario, es decir el valor de este campo coincida con 0, el MAG estará indicando al LMA que el MN que ha provocado el envío de dicho PBU se ha desconectado de su red. El MAG borrará todos los datos relativos al mismo, y el LMA deberá hacer lo mismo, a no ser que reciba otro PBU relativo al mismo MN con un tiempo de vida distinto de 0.

Existen numerosas opciones de movilidad que pueden ser incluidas dentro del mensaje (comentadas en el anexo A) PBU, dónde el orden de ellas no es relevante. Entre todas las opciones posibles que han sido definidas en [4][5], las que se han considerado necesarias para que se ejecute correctamente la funcionalidad esperada de esta implementación, son:

1- CoA: indicará la dirección de acceso global de IPv6 del MAG.
2- Identificador del MN: identificará de manera inequívoca al MN que va a ser registrado. En este caso coincidirá con su dirección HW.

En este caso, ambas opciones tienen el mismo formato, definido en [4] y representado en la siguiente figura:

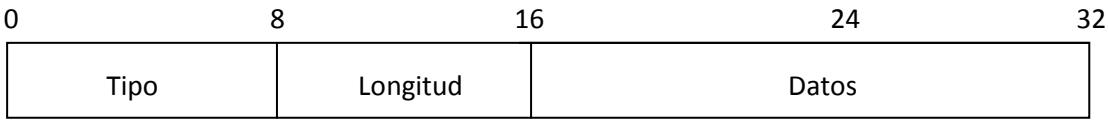


Figura 9: Formato Opciones PBU

El “tipo” es un dato numérico que identifica la opción que viene a continuación. En el caso en el que el destinatario no reconozca ese tipo de opción deberá descartarla y comenzar a procesar la siguiente. La “longitud” corresponderá con el tamaño total de la opción, y los “datos” será propiamente la información necesaria. Es importante saber tanto el tipo como la longitud de cada una de las opciones, para saber cuál es la información que se va a almacenar así como dónde termina esa información para no mezclar varias opciones.

Un ejemplo de un PBU de registro, en esta implementación, será como sigue:

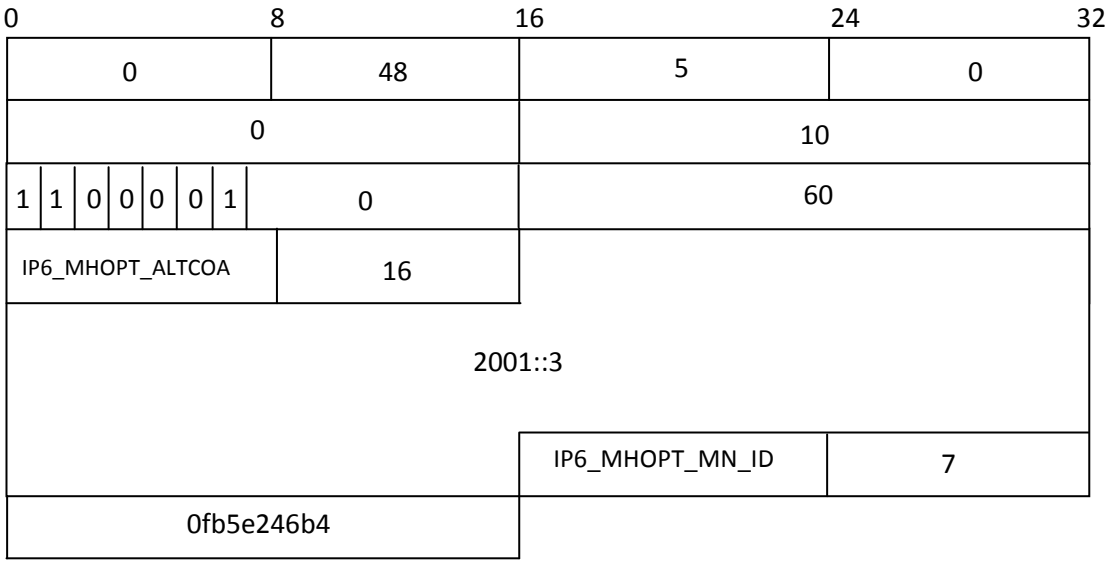


Figura 10: Ejemplo PBU enviado por la implementación realizada

PBA

Cuando en el campo "tipo" de la cabecera de movilidad se encuentra el valor 6, en lugar del 5 como hemos visto hasta ahora, entonces se referirá que a continuación de dicha cabecera de movilidad se encuentra un mensaje del tipo PBA, que será la contestación del LMA al MAG tras recibir un PBU de él. Es necesario que este mensaje sea enviado ya que en el PBU previo se activó el *flag* A.

El formato que le corresponde a este tipo de mensaje, definido en [5], es como se muestra en la *fig. 12*:

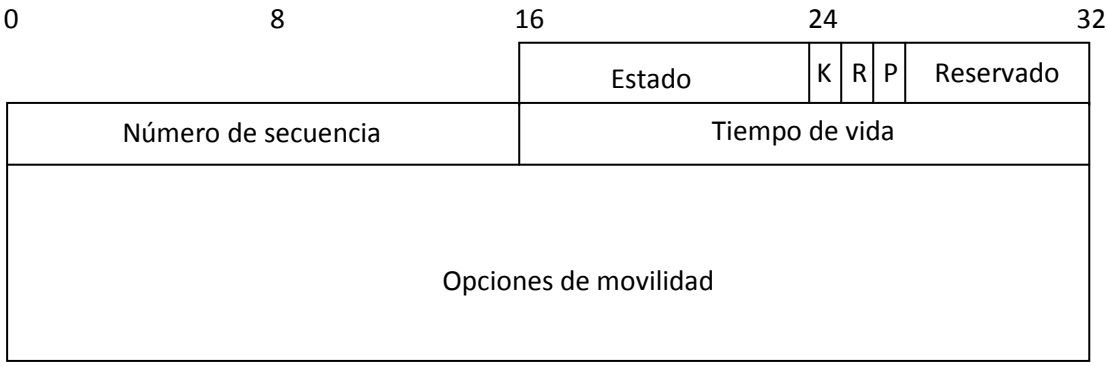


Figura 11: Formato PBA

Dentro de un PBA el campo "estado", corresponde con el estado de aceptación o no de la gestión de la movilidad del MN que la solicita. Además, al ser codificado mediante números, estos también indicarán el motivo por el que haya podido ser rechazado. En el caso de que sea aceptado (un número menor a 128), el código indicará alguna anomalía u

observaciones de su registro. En esta versión reducida de PMIPv6, este campo no es relevante, ya que se ha considerado que si siempre que el LMA recibe un PBU, será porque se le gestionará la movilidad al nodo que lo requiere, y por lo tanto siempre será rellenado con el valor 0.

En cuanto a los *flags*; el bit P significa lo mismo que en el PBU. El *flag* K, es para indicar si se utiliza IPsec en la comunicación entre el MAG y el LMA. Por último, el *bit* R, definido en [12], servirá para indicar si el LMA es capaz de reenviar tanto los mensajes procedentes del MN, como destinados a él. Por tanto, será necesario que sean activados el P y el R.

De la misma manera que antes, el campo "reservado" es únicamente para un posible uso futuro, por lo que será completado con el valor 0.

Igual pasa con el "número de secuencia", que se completa con un número aleatorio, que no se tiene en cuenta para los distintos cálculos para los que suele ser utilizado, al igual que se explicó cuando se comentó este campo en el mensaje PBU.

En el caso de los mensajes PBA, el tiempo de vida nunca será 0. Siempre será establecido a 60. Daría igual el valor que se le asignase dado que el destinatario del PBA ignorará el valor del mismo.

Dentro de opciones de movilidad, y al igual que anteriormente, pueden ser incluidas múltiples previamente definidas. En este caso únicamente se añadirá una opción que coincidirá con el valor del prefijo que le sea asignado el MN que provocó el envío del PBU. Recordar, que en este caso es el LMA el que los asigna, por lo tanto dicha información deberá ser enviada al MAG, para que éste pueda informar al MN del mismo mediante un RA.

Tal y como se define en [5], y a diferencia de lo que sucedía con el PBU, la opción del prefijo tendrá el siguiente formato:

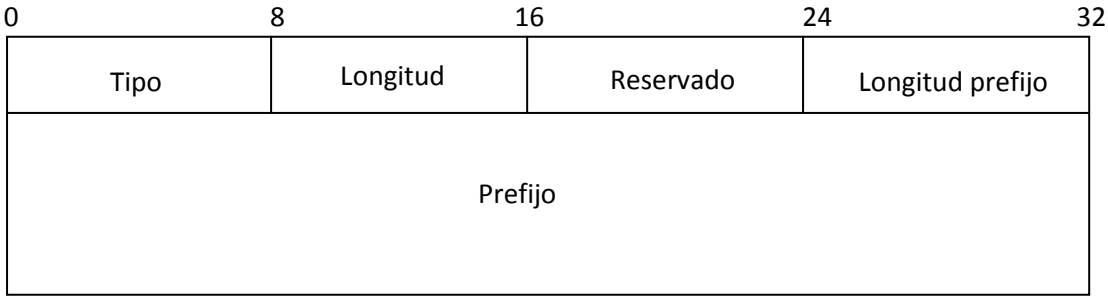


Figura 12: Formato Opción prefijo en PBA

Cabe destacar que únicamente se podrá incluir una opción de este tipo, aunque según [5] puedan ser incluidas múltiples, esta implementación no soporta el hecho de que un mismo MN pueda tener asignados varios prefijos.

Como antes, el campo “tipo” corresponderá con el tipo de opción que va a ser definida a continuación. En este caso la opción sirve para informar del prefijo que se le ha asignado al MN, que además de ser indicado en este campo, se puede reconocer dado que es la única opción que tiene este formato.

El campo denominado “longitud”, contendrá el tamaño total que ocupa esta opción. “Reservado”, al igual que todos los campos anteriores denominados con este nombre será para un uso futuro, pero que en la actualidad únicamente será rellenado con el valor 0, para indicar que el receptor debe ignorarlo.

La longitud del prefijo indicará el tamaño del prefijo que le ha sido asignado al MN. Este campo siempre valdrá 64 debido a las características del MN de nuestro entorno de pruebas, ya que de ser un prefijo de un tamaño distinto éste no será capaz de configurar una dirección IPv6 que comience por dicho prefijo.

El campo “Prefijo” contendrá el prefijo asignado al MN.

A continuación se muestra un ejemplo de un mensaje PBA enviado por el LMA al MAG para asentir el registro de un MN, e informarle de su prefijo.

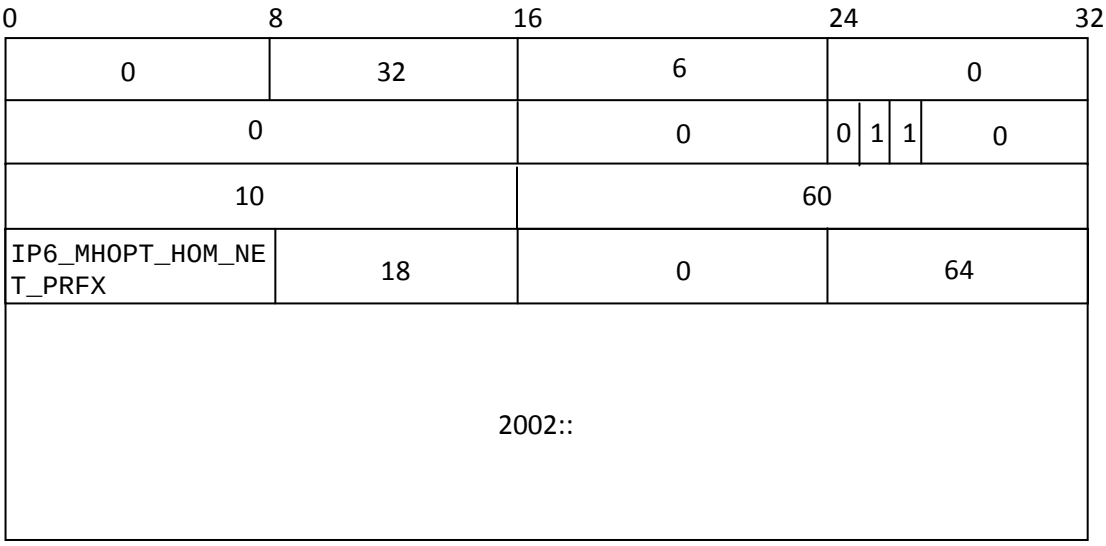


Figura 13: Ejemplo PBA enviado por la implementación desarrollada

RA

Este mensaje es enviado por el MAG al MN, con el fin de que éste último configure una dirección que comience por el prefijo que le ha asignado el LMA de la red. Este mensaje pertenece al protocolo ICMPv6, que se encargará de enviar mensajes de control y de error, entre los que se encuentra el *Echo Request* (ping), entre otros.

Como se puede ver en [13], el formato de este tipo de mensajes es muy simple, dado que es considerado un “subprotocolo” de IPv6.

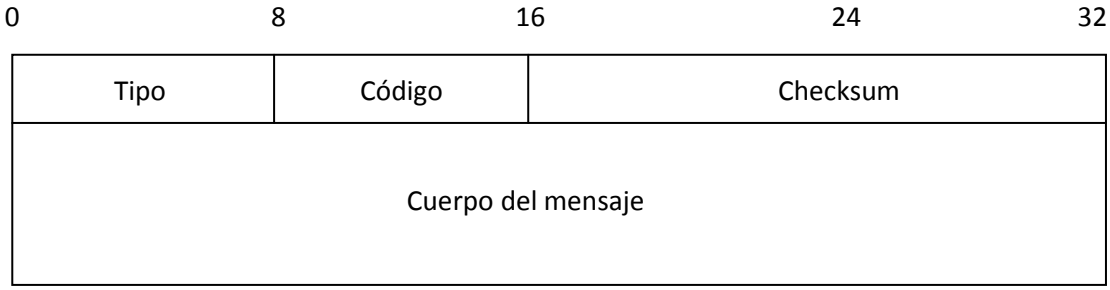


Figura 14: Formato standard de mensajes del protocolo ICMPv6

Como su propio nombre indica, el campo "tipo" revela el tipo de mensaje ICMPv6 que se envía. Además, dependiendo de su valor, también dependerá el formato de los siguientes campos, dado que no se envía la misma información en un mensaje del tipo *echo request* que del tipo RS, o de un mensaje de error. En el caso del RA, el valor de este campo corresponderá con 134.

Asimismo, el campo "código" también dependerá del valor del campo "tipo". Este campo sirve para añadir un nivel adicional de granularidad, que en el caso que nos ocupa, no será necesario, por lo que su valor será 0.

Al igual que en cualquier otro tipo de mensajes, el "checksum" es una suma de comprobación para detectar la presencia de datos corruptos. Al igual que en los mensajes PBA y PBU, esto será realizado por el *kernel* de cada máquina, por lo que será completado con el valor 0 en el programa.

El formato del campo "cuerpo del mensaje", en el caso de un RA, corresponderá con la siguiente figura:

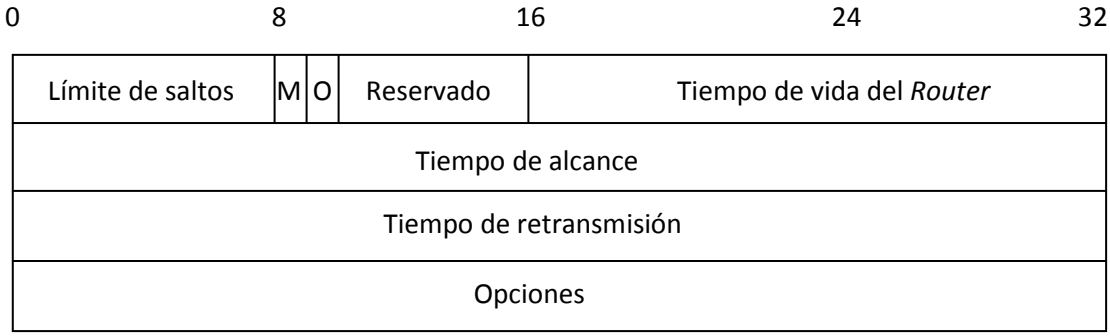


Figura 15: Formato RA

El campo "límite de saltos", establece el máximo de nodos por los que este mensaje puede pasar antes de llegar a su destino.

El *flag* M, en caso de que su valor sea uno, entonces, significará que el nodo que lo reciba deberá configurar su dirección IP utilizando un protocolo de configuración. En el caso

que sea el *bit* O el que valga 1, entonces el nodo que reciba este paquete utilizará el protocolo de configuración para obtener otras características. En este caso, ambos *flags* contendrán el valor 0, ya que como se verá más adelante, en el campo opciones se incluirá información de un prefijo, que por otro lado, es el motivo principal por el que el RA es enviado, es decir, avisar al MN del prefijo que le ha sido asignado.

Como su propio nombre indica, el campo "tiempo de vida del *router*" especificará el tiempo que es válido el RA que se envía. Es decir, el tiempo que el MN puede considerar al MAG como su *router* por defecto, por lo tanto, al que enviará todos los mensajes que quiera enviar por la red. Será establecido con un valor bastante alto, ya que de momento, en esta implementación no se han tenido en cuenta los diferentes tiempos de vida. En este caso, se debería enviar otro RA, cuando este temporizador estuviese a punto de expirar, y siempre y cuando el MN continúe conectado a la red del mismo MAG, para indicar que es a este MAG al que le tiene que enviar todos los paquetes que desee enviar.

En el campo "tiempo de alcance" se informará del tiempo máximo en el que el MN puede considerar "vecino alcanzable" a otro nodo después de recibir la confirmación de que dicho nodo es alcanzable desde el MN.

El "tiempo de retransmisión" indicará el tiempo que el MN debería esperar antes de volver a retransmitir un paquete NS (Neighbor Solicitation). Este tipo de mensajes son utilizados por un nodo para conocer otros nodos que se encuentran en la misma red que ellos. También puede ser utilizado como mecanismo para la detección de direcciones duplicadas.

El campo opciones puede contener diversas opciones, como se especifica en [14]. En este caso, únicamente se añadirá la opción de "Información de prefijo", que indicará al MN que prefijo usar en la red del MAG, es decir, que bits de la dirección identifican a la red.

De acuerdo con [14], el formato de esta opción coincide con el representado en la siguiente figura:

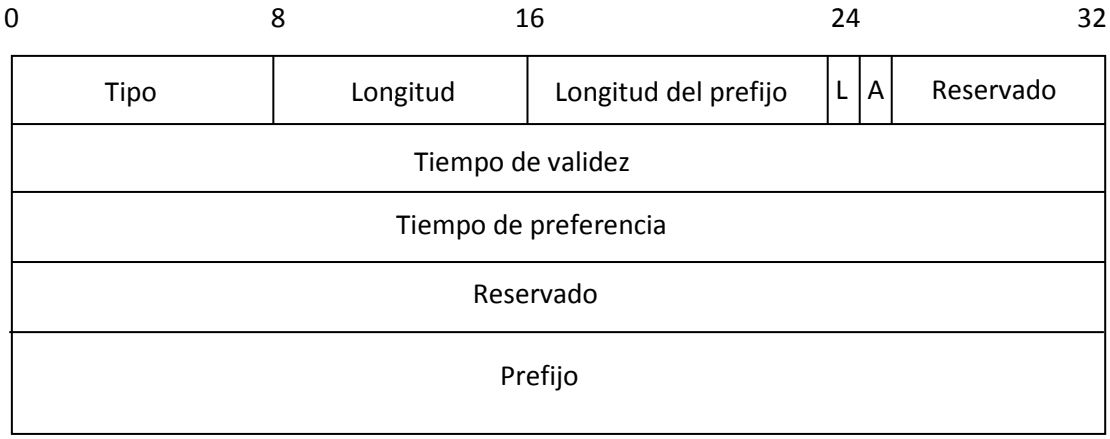


Figura 16: Formato Opción de prefijo en RA

Al incluir esta opción, se indica además que la configuración de la dirección del MN no se realizará mediante el protocolo de configuración DHCPv6, y por lo tanto, los dos *flags*

incluidos en la cabecera del RA deberán ser establecidos a 0. Esta configuración se conoce como “stateless”, lo que permite a un nodo crear su propia dirección IPv6 utilizando diversas variables locales, así como la información obtenida a través del *router*.

El “tipo” de esta cabecera, indicará el tipo de opción que ha sido incluida. En el caso de que la opción sea la de información del prefijo, este campo tomará el valor de 3.

La “longitud” coincidirá con el tamaño de la opción completa. En este caso, dicho tamaño será de 4B. En el caso del campo “longitud del prefijo”, indicará el tamaño del prefijo que la red del MAG tiene como identificador de la misma. En este caso, obligatoriamente, este campo deberá contener el valor 64, ya que de lo contrario el MN de nuestro entorno de pruebas no será capaz de configurar su propia dirección IPv6 para este dominio.

El *bit* L, cuando se encuentre activado, indicará si el prefijo debe ser utilizado para una determinada interfaz. Si por el contrario su valor es 0, no se establecerá que interfaz puede utilizar dicho prefijo, por lo que podrá ser utilizado para cualquier link al que el MN esté conectado o únicamente para uno. En esta implementación ha sido establecido con el valor 1 dado que de esta forma se indica que el prefijo que se le ha asignado únicamente sea utilizado por el link por el cual está conectado a la red del MAG.

El *flag* A, en caso de estar activado, es decir cuyo valor sea 1, indicará que la configuración de la dirección en el nodo que reciba el RA deberá ser del tipo *stateless*. Por tanto, en este caso su valor será uno, ya que como se ha comentado anteriormente será el tipo de configuración que se utilice.

Al igual que en los otros mensajes, los campos “reservados” tendrán un valor 0 dado que han sido reservados para un posible uso futuro, pero que de momento no está siendo utilizado.

El campo “tiempo de validez” establecerá el tiempo en el que el nodo que recibe el RA, en este caso el MN, puede utilizar el prefijo del que se le informa dentro de esa red. Cuando este tiempo esté cerca de vencerse, se le deberá enviar otro RA para actualizar la información. Como se ha explicado con anterioridad, no se tiene en cuenta ningún tipo de temporizador, por lo que este campo será establecido a un valor bastante alto para asegurarse que no expira en el tiempo en el que estemos ejecutando el programa de PMIPv6.

En cuanto al “tiempo de preferencia” indicará el tiempo en el que la dirección que el MN ha configurado a partir del prefijo que se le envía en este RA es utilizada como la dirección preferida del nodo. Destacar, que este valor no debe exceder el tiempo de validez. Al igual que antes, también le será asignado un valor alto, y en este caso el mismo que se le asigna al campo “tiempo de validez”.

En última instancia, irá el valor del prefijo que el LMA asignó al MN. Por tanto, juntando toda la información que se necesita para construir un RA, un mensaje de este tipo, que puede ser enviado por esta aplicación será como lo muestra la siguiente figura.

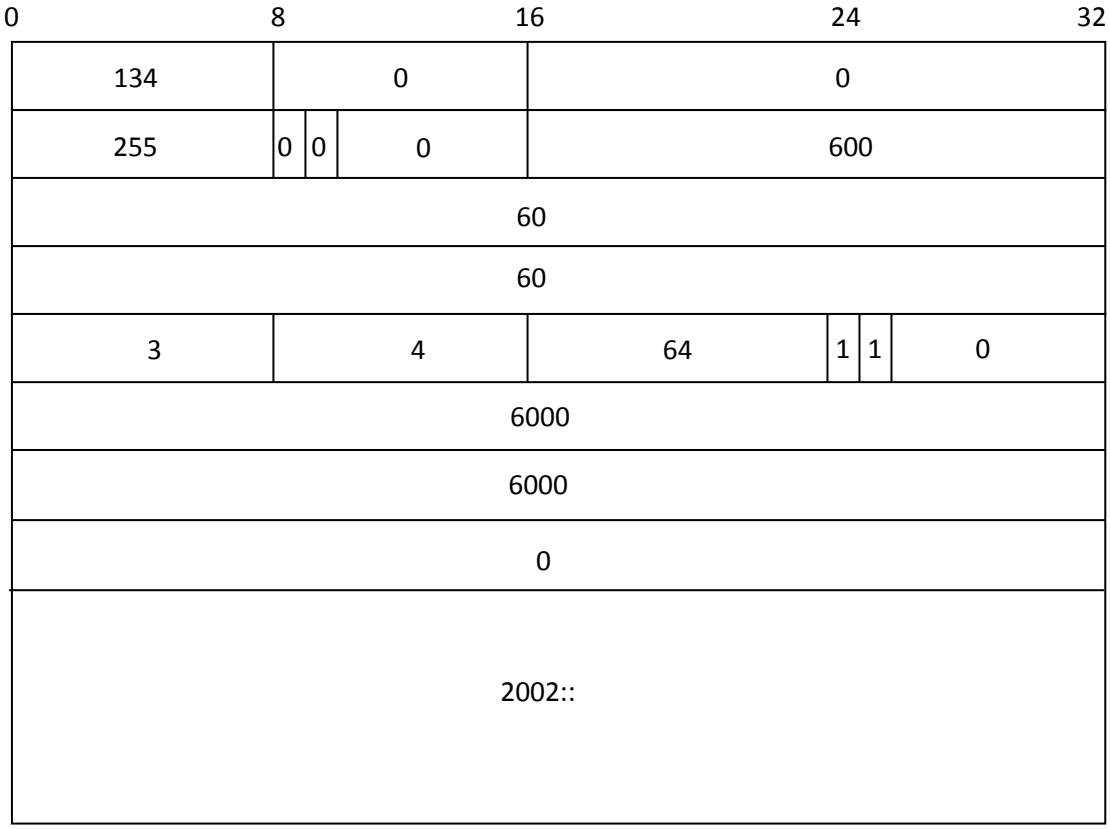


Figura 17: Ejemplo de RA enviado por la implementación desarrollada

4. 4 Estructuras de datos

A lo largo de este apartado se explicaran las diferentes estructuras de datos que han sido utilizadas para el desarrollo de la aplicación. Desde las que han sido creadas, hasta las que el lenguaje de programación C facilita. En cuanto a los mensajes de movilidad se refiere, éstas están detalladas en [15].

Mensajes

Para poder enviar y recibir mensajes a través de *sockets RAW* se ha utilizado la estructura *iovec* definida por C, la cual se encargará de definir tanto la posición de memoria dónde se encuentran los datos que queremos escribir, así como su tamaño en bytes. Esta estructura tiene el formato:

```
struct iovec {
        char *iov_base;
        size_t iov_len;
};
```

Dentro del campo *iov_base* será dónde se incluyan las demás estructuras de datos que son necesarias para la creación de los mensajes. Éstos, como se ha explicado en el apartado anterior pueden estar compuestos por varias cabeceras, y cada una de ellas corresponderá con una estructura distinta, que serán añadidas a esta estructura según el orden del mensaje, es decir de su formato.

Una vez todas las cabeceras de un determinado mensaje hayan sido creadas, y asignado un valor a sus campos, lo que se escribirá en el *socket* será una estructura del tipo `msghdr`, también definida por C. Ésta coincide con:

```
struct msghdr {
        void          *msg_name;
        socklen_t      msg_namelen;
        struct iovec  *msg_iov;
        int            msg_iovlen;
        void          *msg_control;
        socklen_t      msg_controllen;
        int            msg_flags;
};
```

Esta estructura es usada para minimizar el número de argumentos que deben ser pasados a las funciones tanto de escribir como las de enviar mensajes.

Así, el campo *msg_name* contendrá la dirección origen del paquete. También puede contener un puntero nulo si no se desea que se incluya dicho campo. Por tanto, el campo *msg_namelen* contendrá el tamaño de dicha dirección, en caso de que la dirección no haya sido especificada, su valor será 0. En este caso, ambos campos han sido rellenados, utilizando para ello una estructura del tipo `sockaddr_in6`, que sirve para facilitar a un *socket* información acerca de una dirección en un formato de red. Consta de los siguientes campos:

```
struct sockaddr_in6 {
        u_char         sin6_len;
        u_char         sin6_family;
        u_int16m_t     sin6_port;
        u_int32m_t     sin6_flowinfo;
        struct in6_addr   *sin6_addr;
};
```

De esta estructura, se rellenarán todos los campos exceptuando el campo *sin6_flowinfo*, dado que no será necesario. El campo *sin6_len* indicará el tamaño de la estructura. Al trabajar con IPv6, el campo *sin6_family* tomará el valor de *AF_INET6*. El campo *sin6_port* coincidirá con el puerto asignado a lo relativo a movilidad de IPv6. Por último, el campo sin6_addr contendrá la dirección origen del paquete en el formato de red.

Retomando la estructura `msghdr`, el campo *msg_namelen* contendrá el tamaño de la estructura anterior. En el campo *msg_iov* se incluirá la estructura `iovec`, anteriormente mencionada, que contendrá las distintas cabeceras que requiera el mensaje que vaya a ser enviado o recibido. Por tanto, el campo *msg_iovlen* contendrá el tamaño de la estructura asignada al campo *msg_iov*.

Los tres últimos campos de la estructura `msghdr`, no han sido incluidos en los mensajes que son enviados, pero por el contrario, es necesaria su utilización cuando un mensaje es recibido. El campo *msg_control* apuntará a un buffer, que contendrá otra serie de datos relativos al mensaje, denominados datos de control o *ancilliary data*. Como es obvio, la longitud de este buffer será almacenada en el campo *msg_controllen*. En el campo *msg_flags* serán almacenados, como su propio nombre indica, los *flags* que estén activados en el mensaje que haya sido recibido.

Los datos de control serán almacenados dentro de la estructura, también definida por C, `cmsghdr`, cuyo formato es:

```
struct cmsghdr {
          socklen_t   cmsg_len;   /*Data bytes in cmsghdr*/
          int         cmsg_level; /*Originating protocol*/
          int         cmsg_type;  /*Protocol type*/
                                  /* followed by */
          u_char      cmsg_data[];
      };
```

Cada dato de control, o *ancilliary data*, estará almacenado en una estructura de este tipo, obteniendo al final un *array* de esta estructura con los diferentes datos de control que hayan sido incluidos en el mensaje, como puede ser la interfaz por la que ha sido recibido, o diferentes campos de la cabecera IPv6, entre otros. A este tipo de datos solamente se puede acceder utilizando *macros*. Las que han sido utilizadas están descritas en el apartado “Recibir y enviar mensajes” de este mismo capítulo.

Así pues, el campo *cmsg_len* contendrá el tamaño de una estructura de este tipo, es decir de un elemento del *array* de los datos de control. En el campo *cmsg_level* se incluirá el protocolo que creó los datos de control de la estructura en la que nos encontramos. El campo *cmsg_type* almacenará el protocolo que va a leer dichos datos. Por último, en el campo *cmsg_data* se encontrará la información de los datos de control.

Tanto para crear como para almacenar algunos datos de control interesantes para la implementación (en concreto la interfaz por la que se reciben los mensajes), ha sido utilizada una estructura denominada `in6_pktinfo`, a la que le corresponde el siguiente formato:

```
struct in6_pktinfo {
       unsigned int           ipi6_ifindex;
       struct in6_addr        ipi6_addr;
      };
```

En el primer campo de esta estructura se almacenará la interfaz por la que se envía o recibe el mensaje, mientras que el segundo contendrá la dirección destino del paquete. No ha sido necesaria la utilización de más datos de control.

Una vez concluido como está organizado cada uno de los mensajes, se continuará explicando las diferentes cabeceras que pueden formar parte de los mensajes que son enviados en esta aplicación.

Al igual que en la sección anterior, tanto los mensajes del tipo PBU y PBA, obligatoriamente deberán contener una cabecera de movilidad especificando el tipo de mensaje que le sigue. Para implementar esa cabecera, se utilizó la siguiente estructura [15]:

```
struct  ip6_mh {
        uint8_t    ip6mh_proto;
        uint8_t    ip6mh_hdrlen;
        uint8_t    ip6mh_type;
        uint8_t    ip6mh_reserved;
        uint16_t   ip6mh_cksum;
};
```

Cada uno de estos campos corresponde con un campo de la cabecera de movilidad, mostrada en el apartado anterior. El campo que no está incluido en esta estructura, es el que hace relación a los datos propios del mensaje, que serán incluidos mediante otro tipo de estructuras, que se irán añadidos sucesivamente, y por el mismo orden que son explicadas en este documento, en la estructura `iovec`.

Así cuando se trate de enviar un PBU, se utilizará la estructura que se muestra a continuación:

```
struct ip6_mh_binding_update {
        uint8_t     ip6mhbu_proto;
        uint8_t     ip6mhbu_hdrlen;
        uint8_t     ip6mhbu_type;
        uint8_t     ip6mhbu_reserved;
        uint16_t    ip6mhbu_cksum;
        uint16_t    ip6mhbu_seqno;
        uint16_t    ip6mhbu_flags;
        uint16_t    ip6mhbu_lifetime;
};
```

Al igual que antes, esta estructura contiene todos los campos del mensaje PBU, exceptuando las opciones que han sido añadidas por considerarse necesarias. Estos campos son completados con la información explicada anteriormente.

Recordar, que las opciones de movilidad añadidas en los mensajes PBU son; CoA, y el identificador del MN. La primera opción corresponde con la estructura:

```
struct ip6_mh_opt_altcoa {
        uint8_t              ip6moa_type;
        uint8_t              ip6moa_len;
        struct in6_addr      ip6moa_addr;
};
```

En cuando a la segunda opción, la estructura no está definida en [15], por lo que se definió una estructura nueva, con los campos que toda opción de movilidad ha de contener, para ser añadida al mensaje. Ésta está definida por:

```
struct ip6_mh_opt_mobile_node_id {
        uint8_t        ip6mnp_type;
        uint8_t        ip6mnp_len;
        uint8_t        ip6mnp_mnid[7];
};
```

Si por el contrario, se quiere enviar un PBA, entonces la estructura que se ha utilizado, y que sí que está definida en [15], corresponde con:

```
struct  ip6_mh_binding_ack {
        uint8_t   ip6mhba_proto;
        uint8_t   ip6mhba_hdrlen;
        uint8_t   ip6mhba_type;
        uint8_t   ip6mhba_reserved;
        uint16_t  ip6mhba_cksum;
        uint8_t   ip6mhba_status;    /* Status code */
        uint8_t   ip6mhba_flags;
        uint16_t  ip6mhba_seqno;
        uint16_t  ip6mhba_lifetime;
};
```

Igual que sucede con la estructura utilizada en el PBU, ésta contiene algunos campos adicionales, no incluidos en la cabecera, como puede ser el campo ip6mhba_hdrlen, que serán necesarios para poder construir el mensaje. El campo dado como ejemplo, es necesario para determinar el tamaño de la cabecera, y por tanto el tamaño de la estructura `iovec`.

En el caso de los mensajes PBA, únicamente será incluida la opción del prefijo que el LMA ha asignado al MN que ha solicitado el servicio de movilidad. Al igual que sucede con la opción del identificador del MN en el mensaje PBU, la estructura definida para crear esta opción tampoco se encuentra en [15], por lo que ha sido necesario definir una estructura nueva, que corresponde a:

```
struct ip6_mh_opt_hom_net_prefix {
        uint8_t          ip6mnp_type;
        uint8_t          ip6mnp_len;
        uint8_t          ip6mnp_reserved;
        uint8_t          ip6mnp_prefix_len;
        struct in6_addr ip6mnp_prefix;
};
```

Consta de un campo para cada campo de la opción del prefijo que es necesario en un PBA, con el tipo de datos oportuno.

Con respecto a los mensajes RA, también es necesaria la utilización de la estructura iovec, pero en este caso, y como se vio en el apartado anterior no es necesaria la cabecera de movilidad. Sin embargo, será necesaria la cabecera de ICMPv6, la cual corresponderá con la siguiente estructura.

```
        struct icmp6_hdr {
                u_int8_t        icmp6_type;     /*type field*/
                u_int8_t        icmp6_code;     /*code field*/
                u_int16_t       icmp6_cksum;    /*checksum field*/
                union {
                   u_int32_t    icmp6_un_data32[2]; /*specific field*/
                   u_int16_t    icmp6_un_data16[3]; /*specific field*/
                   u_int8_t     icmp6_un_data8[5];  /*specific field*/
         } icmp6_dataun;
} __packed;
```

Esta estructura está mostrada tal y como es definida en la librería “icmp6.h” que también es proporcionada por C. En este caso, únicamente han sido utilizados los tres primeros campos descritos en ella, dado que son los campos propiamente dichos de la cabecera de ICMPv6.

En el caso que nos ocupa, es decir en el mensaje RA, su estructura también es definida en la misma librería que la cabecera de ICMPv6, y es como sigue:

```
        struct nd_router_advert {
                  struct icmp6_hdr        nd_ra_hdr;
                  u_int32_t               nd_ra_curhoplimit;
                  u_int32_t               nd_ra_flags_reserved;
                  u_int32_t               nd_ra_reachable;
                  u_int32_t               nd_ra_retransmit;
                  u_int32_t               nd_ra_router_lifetime;
        };
```

Se puede ver, que el primer campo de esta estructura corresponde con la cabecera ICMPv6, y a continuación los demás campos pertenecientes a un RA. A continuación de éstos se añadirá las opciones que se desee que el RA contenga. En este caso es solamente una, la de información del prefijo, para que el MN sea capaz de configurar una dirección con ese prefijo.

La estructura utilizada también está definida en la librería *icmp6.h* de C. Corresponde con:

```
        struct nd_opt_prefix_info {
                u_int8_t        nd_opt_pi_type;
                u_int8_t        nd_opt_pi_len;
                u_int8_t        nd_opt_pi_prefix_len;
                u_int8_t        nd_opt_pi_flags_reserved;
```

```
        u_int32_t       nd_opt_pi_valid_time;
        u_int32_t       nd_opt_pi_preferred_time;
        u_int32_t       nd_opt_pi_reserved2;
        struct in6_addr nd_opt_pi_prefix;
};
```

Cada campo de dicha estructura pertenece con cada uno de los campos descritos en la cabecera, por tanto para generar dicha opción únicamente habrá que rellenar dichos campos con la información que se precise.

Información almacenada

Para llevar a cabo la funcionalidad especificada, es necesario mantener en cada uno de los equipos cierta información relativa al MN, como puede ser su prefijo, su identificador, etc. Dependiendo del rol que desempeñe cada ordenador (LMA o MAG), mantendrá una información diferente.

Así, en un MAG se almacenarán los datos de los nodos móviles que pueden conectarse a su red, y facilitarles el servicio de PMIPv6, así como los MN que están conectados y se les está gestionando su movilidad. Esta segunda corresponde con lo que en [5] se conoce con el nombre de BUL. En él no se especifica el formato con el que se debe construir. La primera, por el contrario, no está descrita en [5], y será utilizada para almacenar los datos que son leídos del fichero, con información necesaria sobre distintos nodos para poder garantizar su accesibilidad una vez se conecten a la red del MAG, y de este modo poder gestionar su movilidad.

Además, el MAG también deberá almacenar cierta información sobre el mismo para poder establecer comunicación con los demás elementos del entorno de pruebas, como puede ser la interfaz por la que está conectado al LMA. Esta información es leída del fichero de configuración, que ha de ser leído el comienzo de la aplicación, antes de llevar a cabo cualquier otra acción. La estructura que lo almacena es:

```
struct _coa {
        char            coa_addr[INET6_ADDRSTRLEN];
        char            coa_mac[INET6_ADDRSTRLEN];
        char            int_lma_w [50];
        char            int_mn_w [50];
        int             int_mn_n;

};
```

Los datos que se almacenan, en su mayor parte son cadenas, dado que han sido leídos directamente de un archivo de texto. El único campo que no es de este tipo, ha sido transformado de una cadena a tipo “entero” dado que es necesario que sea de ese tipo para ciertas acciones. Como se puede observar, se almacena la dirección IPv6 de acceso global de la

interfaz por la que la máquina en la que la aplicación se está ejecutando está conectada al LMA, y su dirección IPv6 de acceso local por la que se está conectado al punto de acceso al cual se conecta el MN. Estos campos corresponden con el primero y segundo de la estructura respectivamente.

El tercer y cuarto campo, corresponden con el nombre de las interfaces cuyas direcciones son los dos primeros campos, en el mismo orden en el que se ha explicado en el párrafo anterior.

Por último, el quinto campo corresponde con el valor numérico de la interfaz correspondiente al cuarto campo, es decir, la interfaz por la que el MAG está conectado al MN.

Retomando lo anterior acerca de información de los MN, ambas informaciones se almacenarán en listas enlazadas. Cada elemento de la lista, será una estructura diferente.

En el caso de la lista que almacena la información sobre posibles MN a los que se les gestionará su movilidad, cada elemento será del tipo:

```
struct profilentry {
        char                    mn_mac[INET6_ADDRSTRLEN];
        char                    lma[INET6_ADDRSTRLEN];
        char                    hw[9];
        struct profilentry      *sig;
};
```

Todos los datos son del tipo “cadena”, porque son datos leídos directamente de un fichero de texto. La información necesaria, será tanto su dirección IPv6 de acceso local, como si dirección HW para poder utilizarlo como identificador del MN, y la dirección del LMA que vaya a gestionar su movilidad, para que el MAG sepa a quién le debe enviar el PBU de conexión.

En el caso de la lista enlazada correspondiente a la BUL, cada elemento de la lista, es decir cada MN conectado a la red, tendrá un formato como el que sigue:

```
struct bulentry {
        unsigned char           mnID[9];
        char                    prefix[INET6_ADDRSTRLEN];
        struct in6_addr         lma_addr;
        struct in6_addr         coa_addr;
        struct timespec         lifetime;
        int                     tunId;
        struct bulentry         *sig;
};
```

De los nodos conectados, se almacena, su identificador, que coincidirá con su dirección HW.

Además, será necesario conocer su prefijo, para además de comunicárselo, poder crear una ruta para encaminar todos los paquetes con destino el MN. El LMA, es necesario conocerlo, para saber dónde se han de enviar los paquetes procedentes del MN, además de para crear un túnel de comunicación con él. La dirección CoA, que en este caso es la propia máquina en la que se está ejecutando el programa se ha incluido por simplicidad. En cambio, el campo "lifetime", en esta versión del protocolo PMIPv6 no es necesario, pero se ha incluido para facilitar los futuros incrementos de funcionalidad de esta implementación.

Además se almacenará el identificador del túnel que ha sido creado para comunicarse con el LMA del MN, para que pueda ser eliminado una vez que no vaya a volver a ser utilizado. Es decir, que no exista ningún otro MN conectado a la red de ese MAG que necesite los servicios del mismo LMA.

Por otro lado, el LMA almacenará tanto los prefijos de los que dispone para que sean asignados a los diferentes MN que solicitan movilidad, como la *Binding Cache*, que contendrá la información necesaria para garantizar la comunicación con los MN que se hayan conectado. Al igual que sucede con la BUL, en [5], el formato de ella no está especificado, por lo que en diferentes implementaciones de este protocolo podrán ser creadas de formas diferentes.

En el caso de la información de los prefijos que puedan ser asignados, inicialmente es un *array* de estructuras. El tamaño del *array* ha sido limitado a 5, dado que para las pruebas realizadas es un número aceptable. Cada elemento del *array* será una estructura con únicamente dos campos, el prefijo, y otro campo que indique si ese prefijo ha sido o no asignado a otro MN.

En cuanto a la *Binding Cache* respecta, el formato será muy similar al de la BUL. Ésta, también será una lista enlazada en el que cada elemento será una estructura. Dicha estructura, será parecida a la estructura que ha sido creada para la BUL. El formato, de la de la *Binding Cache* es:

```
struct bcacheentry {
        uint8_t                 mn_mac[7];
        char                    prefix[INET6_ADDRSTRLEN];
        struct in6_addr         lma_addr;
        struct in6_addr         coa_addr;
        struct timespec         lifetime;
        struct timeval          last_mod;
        int                     tunId;
        struct bcacheentry      *sig;
};
```

Los datos que se almacenan con respecto al MN son los mismos que se almacenaban en la *Binding Cache*, y además, son utilizados con los mismos fines. El único campo nuevo es el campo relativo a la última modificación. Este campo es utilizado para saber cuándo fue modificada por última vez una entrada de la *Binding Cache*. Es necesario, dado que, en el caso de que un MN cambie de punto de acceso, el LMA recibirá un PBU de desconexión (lifetime =

0), y otro de conexión (lifetime ≠ 0), al recibir el primero, debería eliminar la entrada de la caché, y al recibir el segundo volverla a crear de nuevo. Con este campo, si se recibe un PBU de conexión de un MN que ya está dado de alta en la *Binding Cache*, únicamente se modificará, incluido este campo, y se comprobará, pasado un tiempo prudencial, si después de recibir un PBU de desconexión la entrada de la *Binding Cache* ha sido modificada. Esta comprobación se realizará comprobando si este campo ha cambiado su valor. En caso afirmativo, la entrada habrá sido modificada.

4. 5 Información de entrada

Para conseguir que esta implementación realice correctamente las especificaciones anteriormente descritas, al igual que en la implementación que se probó previamente, son necesarios ciertos parámetros de configuración. Éstos, son introducidos al programa mediante un fichero de texto.

Los datos incluidos en el fichero, denominado "pmip.conf", coincidirá con información relativa a los nodos pertenecientes al entorno de pruebas, como puede ser su dirección IP, las interfaces por las que se encuentran conectados a los otros elementos de la red etc. Esta información es necesaria dado que inicialmente los nodos han de conocer ciertos datos sobre las demás máquinas para poder establecer una comunicación entre ellas.

Para que se pueda leer el fichero, éste se debe encontrar en el mismo directorio en el que se encuentra el ejecutable de esta versión reducida de PMIPv6, y debe ser nombrado exactamente como se ha dicho en el párrafo anterior. Recordar que este fichero será necesario únicamente en los ordenadores a los que se les haya asignado el rol de MAG.

Como se puede observar en la *fig.* 5, lo primero que debe ser realizado al iniciar el programa es la lectura de dicho fichero. Los datos que contenga el fichero serán almacenados en la estructura de datos que corresponda de las que se han comentado en el apartado anterior. Es a partir de entonces, cuando se pueden llevar a cabo las demás acciones, dado que ya se conoce la información de los otros nodos del sistema.

En cada línea del fichero de datos se encontrará el nombre del dato (es decir la variable), a continuación un espacio, y por último el valor que tome esa variable. Contendrá tantas líneas como variables sean necesarias. Este número podrá variar dependiendo del número de MN a los que el MAG en el que se cree pueda prestar el servicio PMIPv6. Cómo se comentará a continuación, por cada MN que se gestione serán necesarias tres variables. El formato del fichero, por tanto, corresponderá con:

VARIABLE VALOR

Entre una línea y otra podrán existir tantos retornos de carro como se desee. El nombre de las variables, al igual que el del fichero, ha de coincidir exactamente con el que se describe en este apartado, y siempre en minúsculas, en caso contrario, serán ignoradas. Las variables que se pueden incluir, serán:

- **coa:** esta variable tomará el valor de la dirección IPv6 de acceso global de la interfaz por la cual el MAG está conectado al LMA.
- **coa_ip:** en este caso, esta variable corresponderá con la dirección IPv6 de acceso local de la interfaz por la que el MAG se encuentre conectado al punto de acceso al que el MN se puede conectar. Recordar, que en ambos MAGs esta dirección debe coincidir, por ese motivo, en uno de los dos MAG fue modificada.
- **interface_coa-mn_num:** esta variable coincidirá con el valor numérico de la interfaz por la que el MAG está conectado al *router*. Es decir, por la interfaz por la que los mensajes destinados al MN han de ser enviados.
- **interface_coa-mn_wor:** el valor de esta variable, será el nombre de la interfaz descrita en la variable anterior.
- **Interface_coa-lma_wor:** como su nombre indica, a esta variable se le asignará el valor del nombre de la interfaz por la que el MAG se encuentre conectado, o por la que se deban enviar los paquetes destinados al LMA.

Las variables anteriormente descritas deberán aparecer en el fichero de configuración una única vez, siendo obligatoria dicha aparición, dado que corresponden con información relativa al MAG, y al ser única no podrá ser repetida. En caso de duplicar una de estas variables, se tomará el valor del último dato escrito en el fichero. Las tres variables que se describen a continuación, podrán aparecer tantas veces como MN se deseen incluir, eso sí, deberán aparecer obligatoriamente las tres juntas, independientemente del orden. Por tanto la información relativa a cada uno de los MN que sean definidos debe ser agrupada. Estos parámetros son:

- **mn_ip:** corresponde con la dirección IPv6 de acceso local de la interfaz por la que el MN se conecta a uno de los puntos de acceso de la red del MAG.
- **mn_hw:** a esta variable le será asignado el valor de la dirección HW de la interfaz por la que el MN se conecta al *router* de la red del MAG. Además, ésta será considerada como identificador de cada uno de los MN, ya que debe ser única, por lo que a dos MN no les podrá corresponder la misma.
- **mn_lma:** en este caso, se almacenará la dirección IPv6 de acceso global del nodo que actúe como LMA, y que sea el que gestione la movilidad del MN que está siendo definido.

En el caso en el que no aparezcan ninguna vez estas variables, el programa interpretará que no se le gestionará la movilidad a ningún MN.

4. 6 Recibir y enviar mensajes

Tanto en la recepción como en el envío de los mensajes pertenecientes al protocolo PMIPv6 que son enviados en esta versión reducida, detallados en el apartado 4.3, se realizan mediante *socket* RAW. En este proyecto, este tipo de *sockets* son utilizados debido a que se desea construir paquetes que incluyan cabeceras estándar, como puede ser la cabecera IPv6, y que además contengan otra serie de cabeceras o campos, distintos a los tradicionales sin que éstos sean modificados por el *kernel*.

En este caso, cada mensaje es construido gracias a la colaboración del *kernel* de cada máquina, dado que se le facilitarán las cabeceras extra, creadas por el programa y será él, el encargado de construir todas las cabeceras restantes para que el paquete pueda ser enviado por la red. Las cabeceras restantes serán las correspondientes desde el nivel IP hasta el nivel inferior de la torre TCP/IP. Esto es debido a que la opción IP_HDRINCL se encontrará desactivada por defecto [16] (y no será activada), por tanto todos los datos que sean escritos en el *socket* son colocados a continuación de la cabecera IPv6.

Para hacer posible que los mensajes puedan ser enviados y recibidos mediante el uso de estos *sockets* es necesario en la creación de los mismos especificar el protocolo de los mensajes que serán recibidos o enviados a través de ellos. Así, cuando el paquete que se desee enviar corresponda con un PBU o un PBA, será necesario especificar que lo que se añadirá a continuación de la cabecera IP corresponderá con una cabecera de movilidad, para que así cuando el *kernel* complete los campos de la cabecera IPv6, en el campo de "cabecera siguiente" especifique que se trata de este tipo de cabera. Esto se consigue especificando el protocolo mediante la constante IPPROTO_MH. El otro tipo de mensajes que son enviados en esta implementación corresponde con RA, y dado que este mensaje se incluye dentro del protocolo ICMPv6, será éste el que será necesario especificar a la hora de crear el *socket* por el que será enviado. La constante que define este protocolo corresponde con IPPROTO_ICMPV6.

Para enviar información adicional en el mensaje, en concreto los denominados datos de control, a los *sockets* se les añaden la opción IPV6_PKTINFO .

En cuanto a la recepción de mensajes, únicamente pueden corresponder con un PBU o con un PBA. El protocolo que es definido en la creación del *socket* corresponderá con el mismo que el del envío, de este modo cuando se reciba un mensaje con cabecera de movilidad, el *kernel* los dirija directamente al programa para que sean tratados por él. Además, para poder acceder a los datos de control, será necesario especificarlo al *socket*. Esto se realiza añadiéndole la opción IPV6_RECVPKTINFO al *socket*. Si se quisiera obtener más información del mensaje, como por ejemplo, la cabecera de enrutamiento también sería necesario especificárselo al *socket.*

Para acceder a este tipo de información será necesario utilizar una serie de *macros* proporcionadas por C, dado que a este tipo de datos no se debe acceder nunca directamente. Las *macros* empleadas en esta implementación son:

- **`CMSG_FIRSTHDR:`** devuelve un puntero a la primera estructura `cmsghdr` en el buffer de datos de control asociado a la estructura `msghdr` facilitada a esta *macro*.
- **`CMSG_NXTHDR:`** esta *macro* devuelve la siguiente estructura `cmsghdr` después de la `cmsghdr` que se le ha pasado como parámetro. En caso de que no existan más estructuras de este tipo en el buffer, devolverá NULL.
- **`CMSG_DATA:`** devuelve un puntero a los datos de la estructura `cmsghdr`.

El proceso utilizado para acceder a este tipo de datos es muy sencillo, dado que simplemente se deberá realizar un bucle, desde la primera estructura `cmsghdr` hasta que no haya más estructuras de este tipo en el mensaje leído. En cada estructura que se recupere, entonces se extraerán los datos que contenga.

En el caso que nos ocupa, el único dato de control que se desea recuperar, y el único añadido, será el correspondiente con la cabecera `IPV6_PKTINFO`, por lo que se comprobará si el dato de control leído coincide con dicho tipo, en tal caso, accederemos a su información.

Una vez recuperada la información perteneciente a los datos de control, la información relativa al propio mensaje ha de ser obtenida, para poder almacenar cada dato en su estructura correspondiente. El mensaje es, por tanto, copiado a una estructura del tipo del mensaje correspondiente, es decir en caso de que el mensaje leído se trate de un PBA la estructura en la que se copiará será `ip6_mh_binding_ack`, y en el caso de recibir un PBU la estructura en la que será copiado coincidirá con `ip6_mh_binding_update`. De este modo, los datos quedarán almacenados en su campo correspondiente en la estructura pertinente, por lo que su recuperación es trivial.

En cuanto a las opciones incluidas en los diferentes mensajes, se almacenarán en un *buffer* de caracteres. Dado que el formato de estos mensajes es conocido, es decir; el orden de las opciones siempre será el mismo, los diferentes campos de los que conste cada opción etc. Se recorrerá identificando a qué información requerida pertenece cada dato que sea leído de dicho *buffer*, y será almacenado en el lugar correspondiente.

4. 7 Creación de rutas y túneles

En cuanto al establecimiento de la comunicación entre los diferentes nodos del entorno de pruebas se refiere, como se ha ido comentando a lo largo de este capítulo esto es realizado mediante llamadas al sistema. Éstas son realizadas mediante la función de C *system()*. A esta función se le pasa como parámetro una cadena, que coincidirá con el comando que debiera ser ejecutado en la consola de Linux.

Para asegurar que dos nodos se puedan comunicar, será necesario hacer uso de esta función una serie de veces a lo largo del programa. El número de veces, y cuando se realicen dependerá del ordenador en el que se esté ejecutando el programa (LMA o MAG). Las acciones llevadas a cabo por esta función son seis: crear/eliminar rutas, crear/destruir/modificar túneles y crear vecinos. Los comandos necesarios para realizar las dos primeras acciones respectivamente, corresponde con:

 # /sbin/ip -6 route add *dirección-destino* dev *interface*
#/sbin/ip -6 route del *dirección-destino* dev *interface*

Para la creación de un túnel es necesaria la utilización de más comandos, con lo que ello conlleva, más llamadas al sistema. Primeramente se deberá crear el túnel propiamente dicho, a continuación se deberá activar el dispositivo que se acaba de crear, y por último será necesario asignarle una dirección, que en este caso coincide con la dirección IPv6 de acceso global que tenía asignada la interfaz en la que se crea el túnel. Estas tres acciones corresponden cada una de ellas con un comando:

#/sbin/ip -6 tunnel add *nombre-túnel* mode ip6ip6 remote *dirección-final-túnel* local *dirección-origen-túnel* dev *interface*
#/sbin/ip link set *nombre-túnel* up
#/sbin/ip addr add *dirección-coa* dev *nombre-túnel*

A la hora de eliminar o de modificar el final del túnel únicamente será utilizado un comando para cada una de estas acciones, que serán respectivamente:

#/sbin/ip -6 tunnel del *nombre-túnel* mode ip6ip6 remote *dirección-final-túnel* local *dirección-origen-túnel* dev *interface*
#/sbin/ip -6 tunnel change *nombre-túnel* mode ip6ip6 remote *dirección-final-túnel* local *dirección-origen-túnel* dev *interface*

Cuando se quiera crear un nodo vecino, en la tabla de vecinos, se deberá ejecutar el siguiente comando:

#ip -6 neighbor add *dirección-local-nodo-vecino* lladr *dirección-hw-nodo-vecino* dev *interface*

Así, en el caso en el que nos encontremos en un MAG, para que éste se pueda comunicar con los otros elementos del entorno, MN y LMA, será necesario realizar todos los comandos previamente descritos. Cuando le llega el aviso de que un MN se ha conectado a su red, entonces éste debe enviar un PBU al LMA, y cuando recibe el PBA del LMA, creará un túnel de comunicación con él, creará la ruta para que los mensajes con destino al LMA se envíen a través de ese túnel. A continuación, se crea la ruta, para que todos los paquetes destinados a una dirección que comience por el prefijo que se le ha sido asignado al MN se envíen por la interfaz por la que está comunicado el MAG con el MN. Por último, antes de

enviar el RA al MN, el MAG se asegurará que éste es su vecino creando una nueva entrada en la tabla de vecinos mediante el comando explicado.

Cuando se trata de eliminar, las acciones se llevarán a cabo inmediatamente después de que el MAG envíe el PBU de desconexión al LMA. Primero se borrará la ruta creada hacia el MN, y a continuación el túnel de conexión con el LMA. No será necesario eliminar la ruta creada hacia el LMA, ya que al desaparecer este dispositivo, desaparecen todas las rutas que se encaminasen a través de él.

Por el contrario, si el programa se ejecuta en el LMA, dependiendo de la situación, se podrán ejecutar todos los comandos previamente expuestos excepto en el que se crea un nuevo vecino. Cuando se recibe en este nodo un PBU de conexión de un MN cuya entrada en la *Binding Cache* no exista, se ha de crear, tanto la ruta como el túnel de comunicación con el MAG. Además, posteriormente a asignarle un prefijo al MN, se debe crear una ruta hacia él a través del túnel. En el caso en el que el PBU sea de conexión y exista una entrada de ese MN en la *Binding Cache*, únicamente se procederá a modificar el final del túnel. No es necesario realizar ninguna acción más debido a que todos los mensajes relativos a PMIPv6 en este nodo son encaminados a través de este túnel, y dichas rutas continuarán siendo válidas.

Solamente existe una circunstancia en la que se ejecuten las acciones de eliminación. Cuando se recibe un PBU de desconexión, y pasado un tiempo prudencial, no se ha recibido un PBU de conexión relativo al mismo MN, entonces, se procederá a eliminar el túnel que une a este LMA con el MAG correspondiente (siempre y cuando no sea utilizado por otro MN). Al igual que sucedía en el caso anterior, no será necesario eliminar las rutas que utilizaban dicho túnel, ya que al eliminar el túnel dichas rutas desaparecen de la tabla de rutas automáticamente.

4. 8 Construcción del archivo ejecutable

Para generar los dos archivos ejecutables de PMIPv6 y compilar el código fuente ha sido utilizado un archivo *makefile*. En el cual, se incluyen únicamente la información necesaria para crear el ejecutable, y dos acciones.

La información denominada necesaria coincide con el nombre de los objetos a crear, es decir las diferentes librerías que se han creado para la implementación del programa (de cada una de ellas se tendrá un archivo *.h* y otro archivo *.c*), el nombre que recibe el archivo ejecutable, que en nuestro caso ha sido denominado *pmipv6*, y los *flags* que son necesarios, como puede ser - *lpthread* para que se puedan crear nuevos hilos de ejecución.

Las acciones que se han incluido son las de crear el archivo ejecutable, con sus correspondientes archivos objeto, y el de eliminarlos.

Por tanto, para crear el archivo ejecutable del programa, se deberá ejecutar el siguiente comando, dentro del directorio dónde se encuentren todos los archivos del código fuente y el archivo *makefile*:

#make

De esta manera, ya se podrá ejecutar el programa, como se ha mencionado anteriormente, y dentro del mismo directorio mediante el comando:

#./pmipv6

En el caso que se quiera eliminar todos los archivos objeto (*.*o*) y el archivo ejecutable, en ese mismo directorio se ejecutará:

#make clean

CAPÍTULO

5

VALIDACIÓN Y EVALUACIÓN

El hombre puede creer en lo imposible, pero nunca creerá en lo improbable.
Oscar Wilde (1854-1900)

5. 1 Introducción

Una vez desarrollada la implementación de una versión reducida de PMIPv6, es necesaria la comprobación de su correcto funcionamiento. Para ello, se establecen una serie de características que han de ser probadas. En el caso en el que los resultados obtenidos sean los esperados, se podrá asegurar que el programa funciona como se desea.

Inicialmente, y durante el proceso de desarrollo de la aplicación, han sido probadas todas las funcionalidades que son necesarias para que el protocolo sea llevado a cabo satisfactoriamente. Entre ellas se incluirían por ejemplo, el registro de un nuevo MN tanto en la *Binding Cache* del LMA como en la BUL del MAG, o su eliminación de ambas, o la lectura del fichero de configuración, entre otras.

Recordar, que esta nueva implementación, consiste en dos programas que son ejecutados por separado, en cada ordenador se ejecutará el programa que corresponda con el rol que va a ejecutar. Si bien, ambos programas han sido validados en la fase de desarrollo, ahora es necesario comprobar que al ejecutar los dos en las tres máquinas (2 MAG, y 1 LMA), la comunicación entre ellos coincide con el orden de la señalización de PMIPv6, que los mensajes enviados son los correctos, y que el MN es accesible tras diversos cambios de punto de acceso, independientemente del tiempo que se encuentre en cada uno de ellos.

Dado que en ciertas aplicaciones el tiempo en el que el MN se queda incomunicado, es decir, no es accesible por ningún otro nodo de la red, es de vital importancia, dependiendo de la función que se quiera realizar, se ha considerado oportuno realizar un análisis de dicho tiempo. Éste será realizado mediante una toma de datos repetidamente para así, poder extraer una conclusión de los mismos.

5. 2 Entorno de pruebas

El escenario básico en el que se llevarán a cabo las pruebas del programa que implementa PMIPv6 constará de cuatro ordenadores, uno de ellos portátil (MN), y dos puntos de acceso (*routers* configurados como puntos de acceso). Todos estos elementos se conectan como indica la siguiente figura.

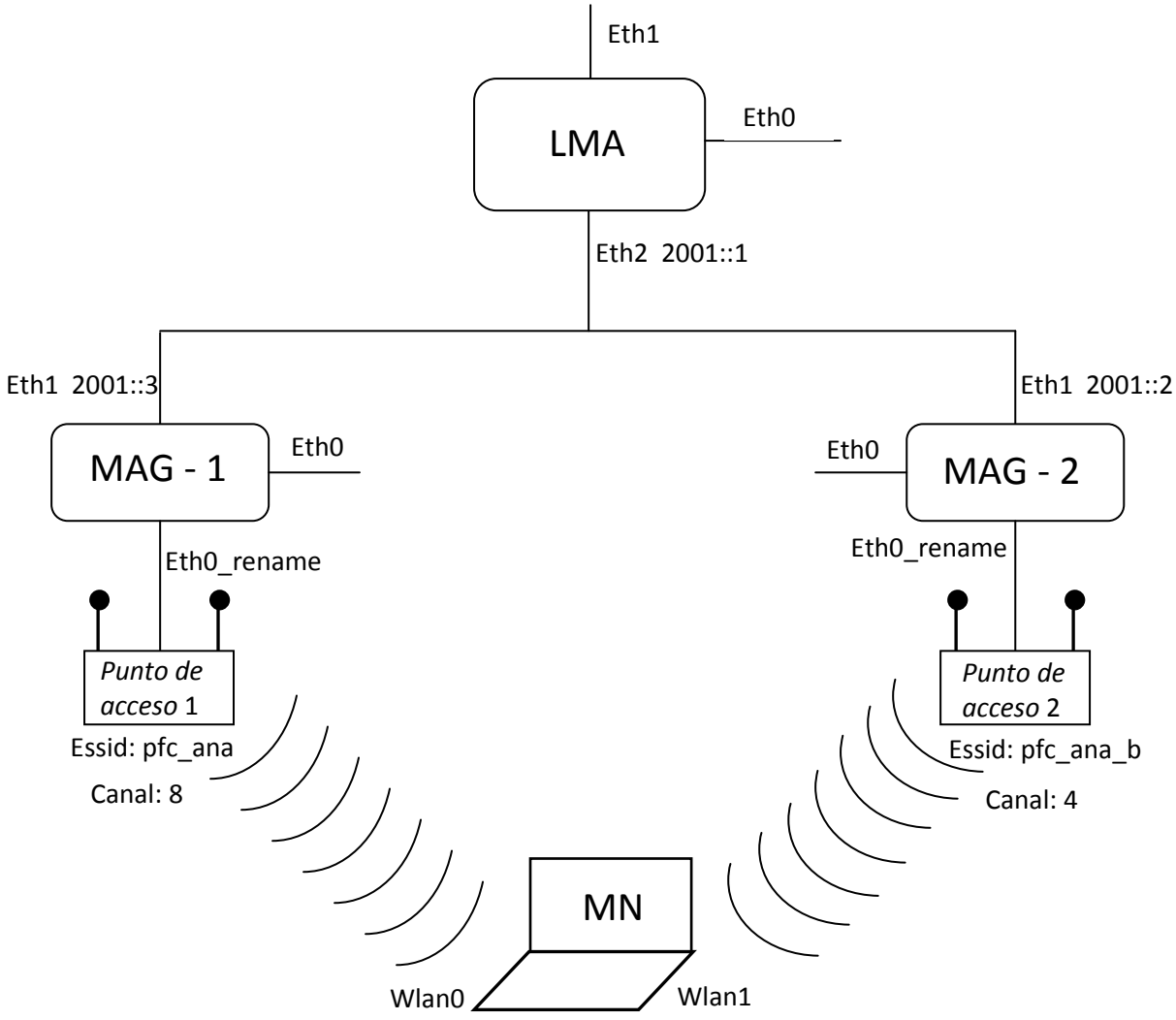


Figura 18: Entorno de pruebas

Se puede observar que todos los ordenadores, a excepción del nodo móvil (portátil) constan de tres interfaces. En todos ellos existe la interfaz "eth0", la cual, en las primeras pruebas, no será utilizada dado que es la interfaz por la que estos ordenadores están conectados a la red del laboratorio.

5. 3 Validación del funcionamiento

A lo largo de este apartado, se describirán las características que han sido probadas en la implementación para, efectivamente, poder asegurar que su funcionamiento es el esperado, además de aportar una estabilidad importante, debido al restablecimiento de la comunicación con el MN.

El objetivo de la validación del funcionamiento, es que el programa desarrollado (en este caso dos), cumpla con todos los requisitos especificados, que su interacción sea la esperada, y que como consecuencia de las dos anteriores, la funcionalidad del mismo se desarrolle satisfactoriamente. Es en este proceso donde se llevarán a cabo lo que se conoce como pruebas funcionales.

Tras la validación de cada programa por separado, este apartado se centrará en la comprobación que el objetivo final del protocolo es cumplido, es decir, que el MN es accesible por cualquier otro nodo, en cualquier situación. Por lo tanto, las diferentes características que serán comprobadas serán:

1. La señalización es la correcta. El término correcta hará referencia a que coincidirá con la señalización representada en las figuras *fig. 2* y *fig. 3*. Por tanto, será necesario comprobar que:

    a. MAG envía PBU de conexión al LMA cuando un nodo se conecta a su red. El tiempo de vida de dicho PBU deberá ser distinto de 0.
    b. LMA recibe PBU de conexión, asigna un prefijo al MN (siempre que sea la primera vez que dicho nodo solicita movilidad), y confirmará el registro de dicho MN al MAG mediante un PBA.
    c. El MAG recibe el PBA, y registra el MN en su BUL.
    d. MAG envía un RA al MN informándole que él es su *router* por defecto, y del prefijo que le ha sido asignado, con el fin de que éste configure una dirección que comience con dicho prefijo.

2. Comprobar que tras un cambio de punto de acceso del MN, el MAG del que se desconecta envía un PBU de desconexión. Es decir, el campo tiempo de vida debe ser igual a 0. El MAG al que se conecta deberá llevar a cabo la señalización detallada en la prueba 1.

3. Al realizar un cambio de punto de acceso, al MN se le volverá asignar el mismo prefijo, y éste configurará la misma dirección IPv6 de acceso global. Con lo cual, uno de los principales objetivos de PMIPv6 se cumpliría.

4. Tras la señalización pertinente, se ha de comprobar que efectivamente, tanto las rutas necesarias, como el túnel han sido creados en todos los nodos entre los que se ha llevado a cabo dicha señalización.

5. Será necesario además, comprobar que la comunicación con el MN se establece la primera vez que se conecta a la red de un MAG, al igual que queda restablecida cuando se cambia de punto de acceso a la misma red.

Para llevar a cabo estas comprobaciones, ha sido necesario crear un entorno de pruebas, tal y como se ha documentado en el apartado 5.1 de este mismo capítulo, dónde se especifican tanto los nodos como las operaciones que han sido necesarias realizar en ellos para crearlo.

La metodología de las pruebas será sencilla, puesto que únicamente habrá que ejecutar cada programa en el ordenador correspondiente, a continuación ejecutar el programa que se ejecuta en los puntos de acceso del entorno de pruebas (ambos comentados en el capítulo 4 y en el anexo E respectivamente). Una vez realizado esto, no será necesaria ninguna acción más en ninguno de los dispositivos mencionados, únicamente habrá que simular el movimiento del MN. Para ello se deben llevar a cabo lo siguientes pasos:

- La interfaz inalámbrica estará desactivada inicialmente, por lo que será necesario activarla, para ello con permisos de usuario *root*, ejecutamos el siguiente comando:

  #ip link set wlan0 up

  Dónde, “wlan0” corresponde con la interfaz inalámbrica que deseamos activar.

- A continuación, nos conectamos a la red de un MAG. Esto se consigue mediante el comando:

  #/sbin/iwconfig wlan0 essid pfc_ana

  Recordar que por motivos de simplicidad se denominó a la red de cada *router* de una manera diferente. Así al *router* conectado al MAG-1 le corresponde la red “pfc_ana” y al *router* conectado a la red del MAG-2 le corresponde la red “pfc_ana_b”.

- Independientemente del tiempo que el MN haya permanecido conectado a la red de un determinado MAG, se conecta a la red del otro MAG. El comando utilizado será el mismo que el anterior, pero modificando el *essid*, es decir, ir alternándolos.

- El proceso de cambio de punto de acceso en el MN se repite un número indeterminado de veces, esperando a realizarlo cada vez un período de tiempo

distinto, para comprobar que la comunicación se restablece independientemente del tiempo que se haya permanecido conectado a dicho punto de acceso.

Para comprobar que, efectivamente, el MN vuelve a ser accesible, en él se ejecutará, como se ha comentado con anterioridad, el comando *ping6* con destino la dirección IPv6 de acceso global del LMA. Mediante este comando, se comprobará que se podrán recibir paquetes del LMA, quedando interrumpida dicha recepción durante un período de tiempo cuando el MN cambia de punto de acceso de conexión a la red, pero transcurrido dicho período, se vuelve a observar, en el MN, que se reciben paquetes procedentes del LMA.

Además, para observar tanto los mensajes enviados entre el LMA, el MAG y el MN como su contenido, se utilizará, como se ha comentado el *sniffer Wireshark*. Serán utilizados capturas de este programa para demostrar el correcto funcionamiento de la aplicación. Además también será mostrado que, como consecuencia de la señalización se crean las rutas y túneles necesarios. De esta manera, el funcionamiento del programa queda demostrado, dado que lo que es necesario que se realice, es ejecutado, y además con éxito debido a que el objetivo también es alcanzado.

Una vez realizadas todas las comprobaciones comentadas anteriormente, además se realizan dos nuevos casos de prueba, para cerciorar que, no sólo el MN es accesible por otros elementos de la red, sino que también otras funcionalidades, propias de cualquier ordenador conectado a una red, pueden ser llevadas a cabo. En este caso, las aplicaciones probadas con:

- Conexión SSH desde el MN al LMA. Dicha conexión se establece la primera vez que el MN se conecta a la red de cualquiera de los dos MAG pertenecientes al dominio de pruebas, y se comprueba que tras varios cambios de punto de acceso, la conexión al LMA se mantiene, pudiendo ejecutar en el MN cualquier operación, y cuyas consecuencias pueden verse reflejadas en el LMA.
- Emisión de un video desde el LMA al MN. Gracias al reproductor VLC, un ordenador puede emitir un archivo de vídeo que tenga almacenado a otro ordenador en la misma red. La prueba a realizar, en este caso, será que el LMA emita un video al MN y que éste sea capaz de recibirlo y reproducirlo, mientras va cambiando de ubicación dentro de la misma red. Tras realizar la prueba se puede observar que el video se reproduce en el MN correctamente, cuando se produce un cambio de punto de acceso a la red, la reproducción del video es interrumpida, volviendo a reanudarse al poco tiempo, es decir, tras el tiempo de incomunicación.

La siguiente tabla contendrá toda la información de direcciones IPv6, relativa a los nodos utilizados en el entorno de pruebas para un mejor entendimiento de las pruebas realizadas. En el caso del MN no se dispondrá de una dirección IPv6 de acceso global hasta que se le haya asignado un prefijo.

ROL	IPv6 ámbito global	IPv6 ámbito local
LMA	2001::1	No utilizada
MAG-1	2001::3	fe80::227:19ff:feb0:1de
MAG-2	2001::2	fe80::227:19ff:feb0:1de
MN	-	fe80::20f:b5ff:fee2:46b4

Tabla 1: Resumen de nodos y direcciones IPv6

5. 4 Pruebas funcionales

A continuación se muestran todas las pruebas recogidas para verificar que el funcionamiento de la aplicación desarrollada es el esperado.

Como evidencia a la primera característica que necesita ser probada, se mostraran dos capturas de *Wireshark*, una en el LMA, y la otra perteneciente al MAG, para comprobar que efectivamente la señalización se produce, tal y como lo específica el protocolo PMIPv6 [5]. En el caso del MAG será necesario mostrar dos, porque los diferentes mensajes que debe enviar, son enviados por diferentes interfaces.

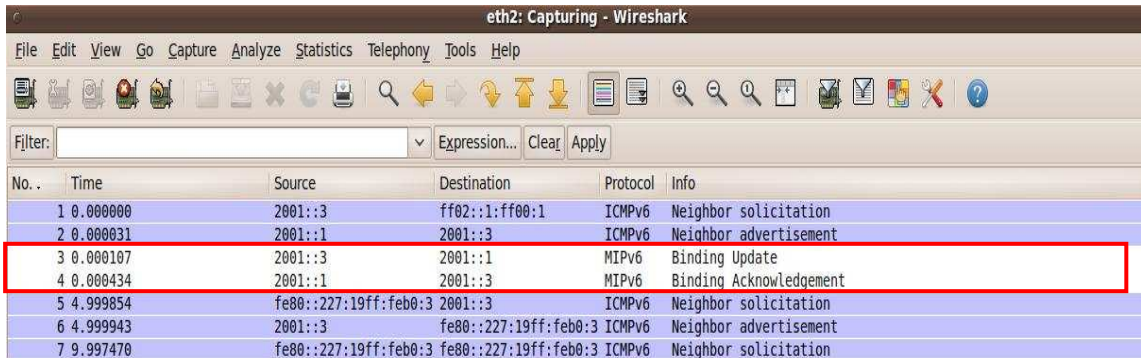


Figura 19: Secuencia de mensajes recibidos en LMA

En el caso de la interfaz por la cual, el MAG se encuentra conectado al LMA, la secuencia de mensajes será exactamente la misma, por lo que no será mostrada. En cambio, si se mostrará el momento en el que él envía el RA al MN.

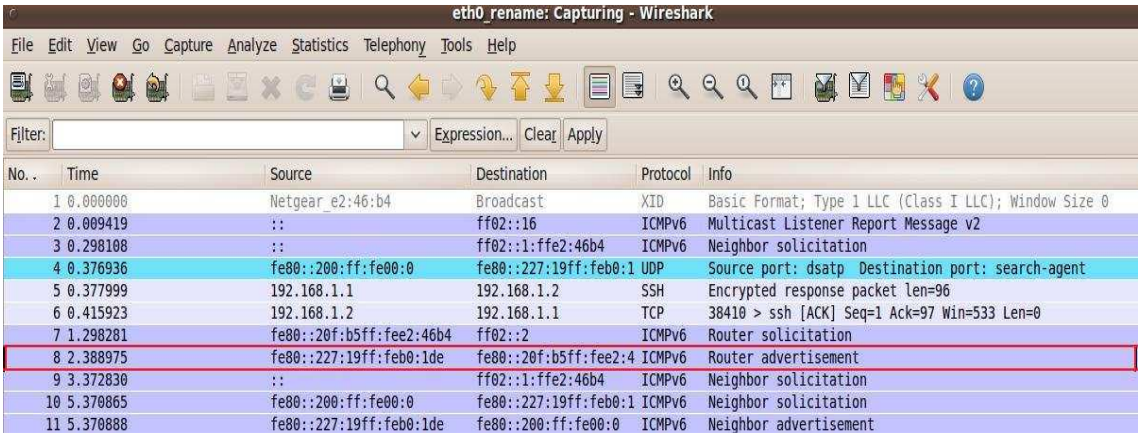


Figura 20: Secuencia de mensajes en MAG-MN

A continuación se muestra el contenido de los tres mensajes enviados gracias a la aplicación desarrollada. Son mostrados en el mismo orden que son enviados.

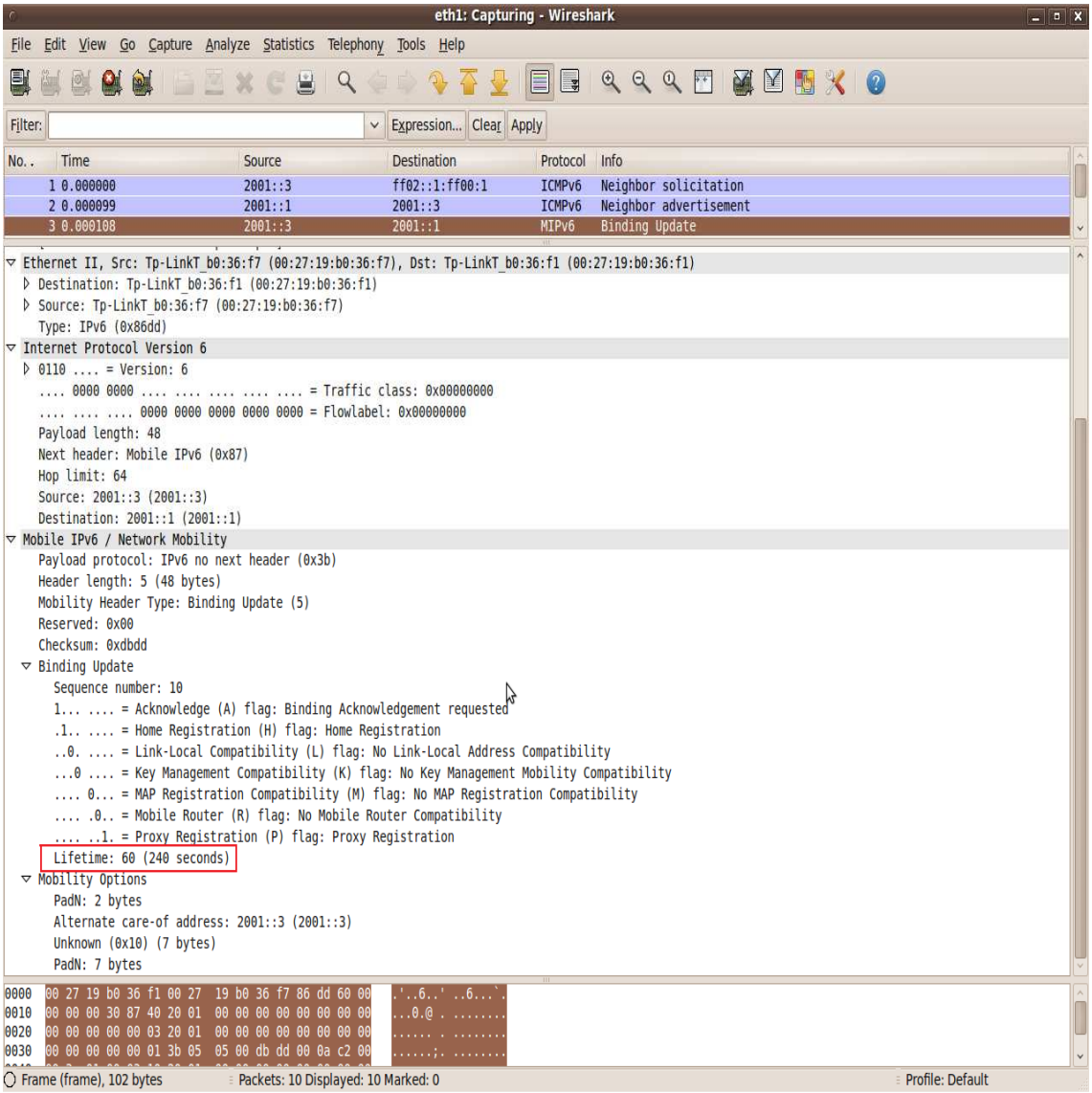


Figura 21: Mensaje PBU de conexión MAG-LMA

Este es el primer mensaje enviado en la señalización del protocolo PMIPv6. Se envía al LMA correspondiente, una vez que un MN se ha conectado a la red. Se diferencia este PBU, de un PBU de desconexión únicamente en el tiempo de vida, que como se puede observar en este caso es de 60, mientras que como se verá más adelante, en un PBU de desconexión es de 0.

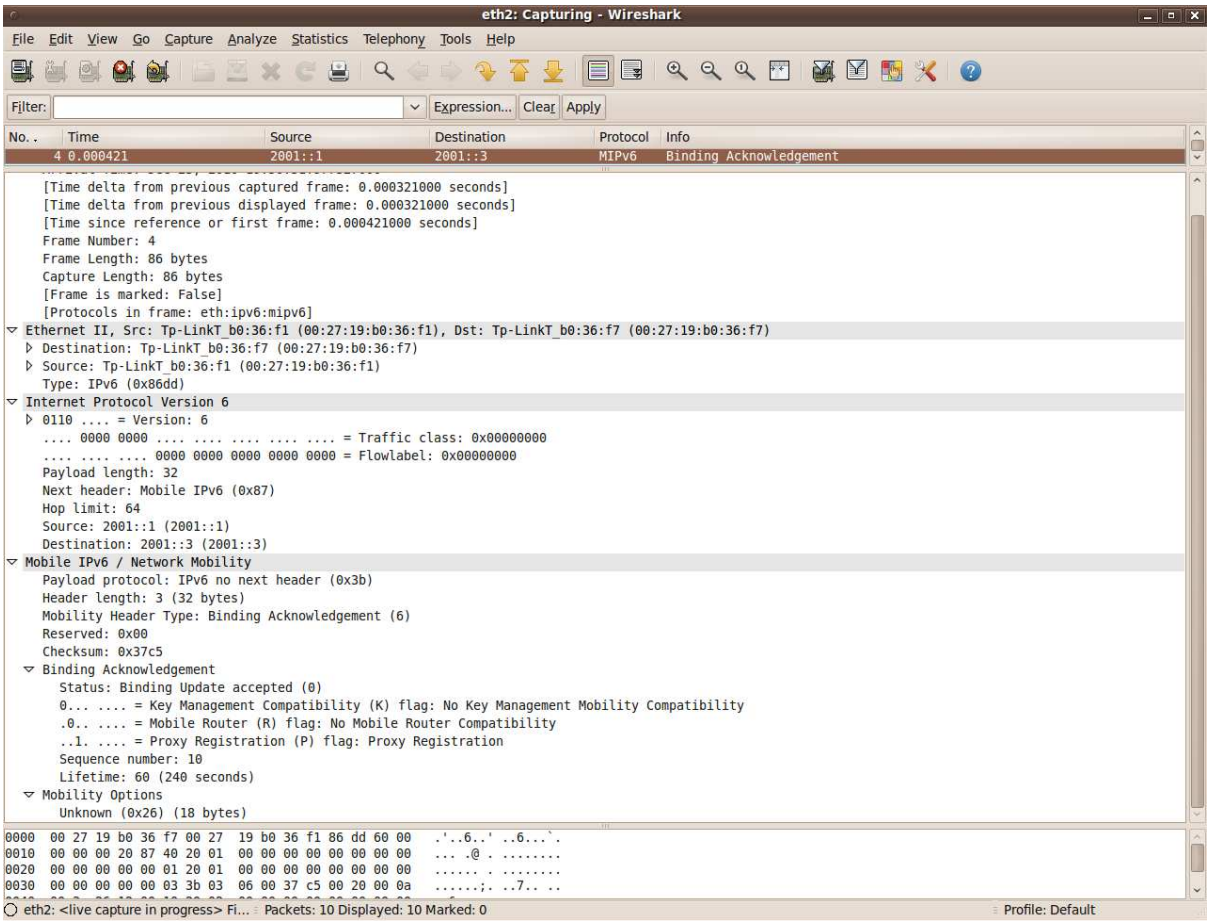


Figura 22: Mensaje PBA LMA-MAG

Este es el mensaje que envía el LMA como contestación el PBU, asintiendo el registro del MN. En él se encontrará la información del prefijo que éste le ha asignado al MN. Esta información se incluye en el campo de *Mobility Options*.

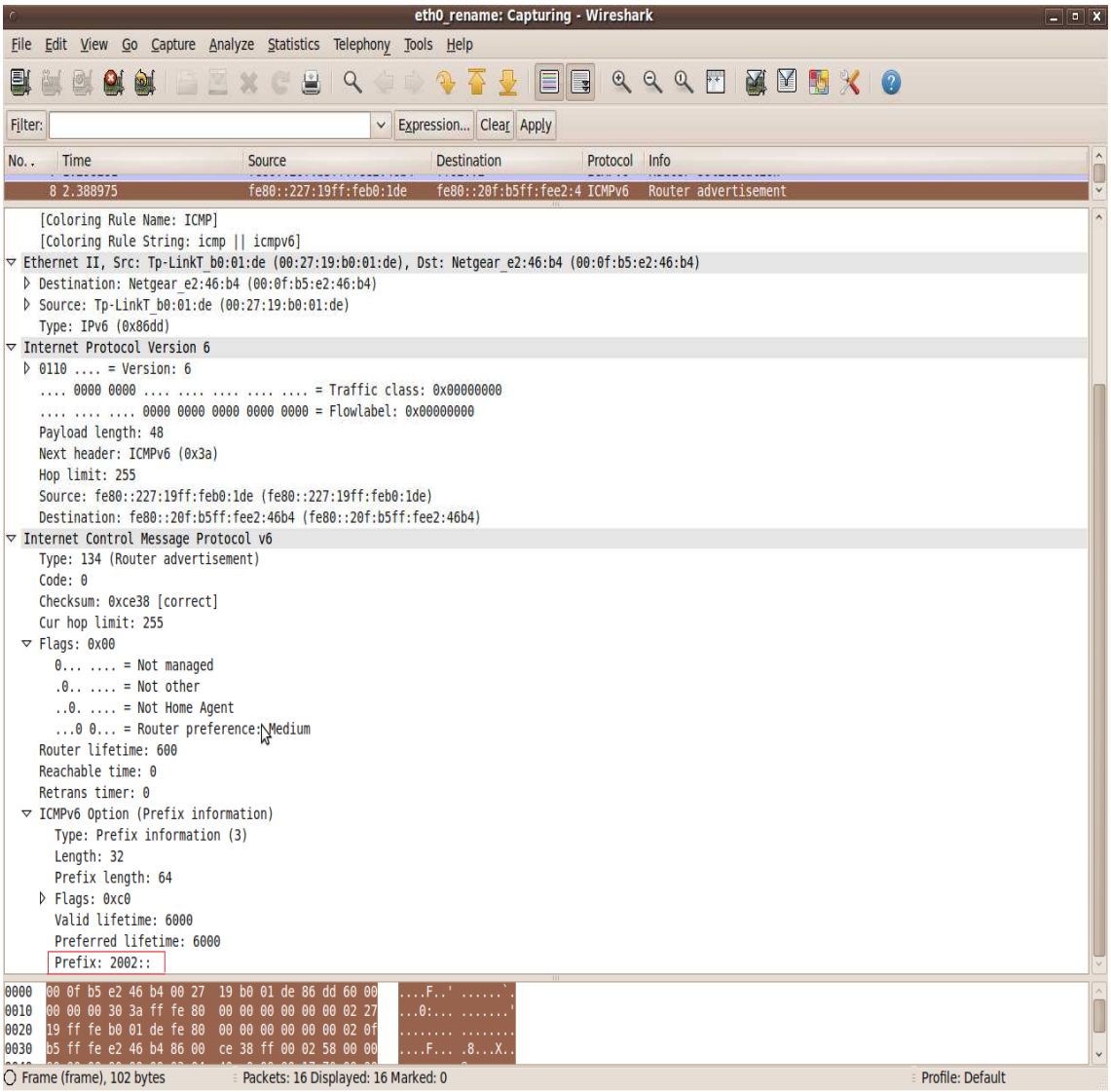
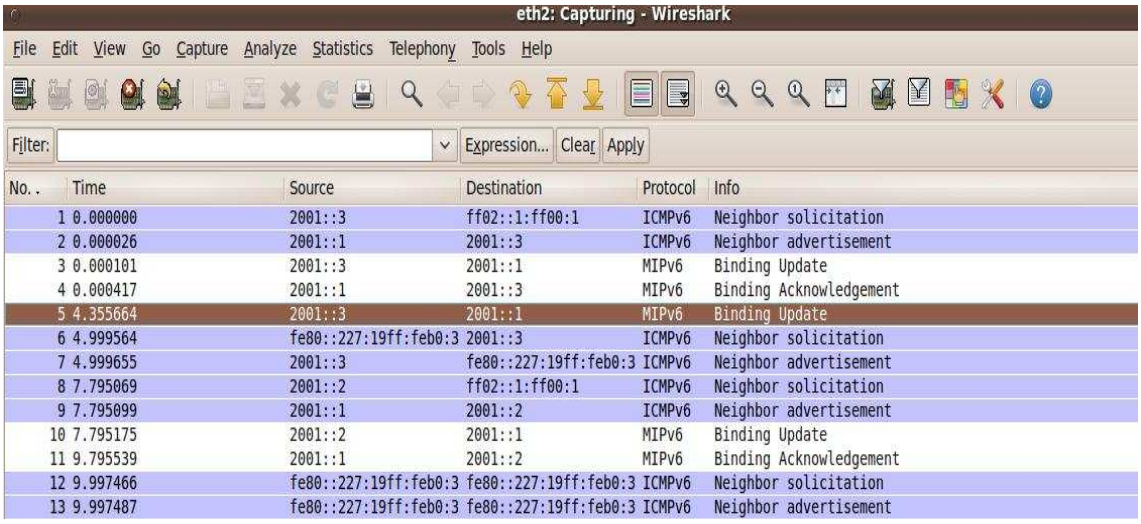


Figura 23: Mensaje RA MAG-MN

En este mensaje, se puede observar como el prefijo asignado al MN que ha solicitado la movilidad, en este caso es “2002::”.

Cuando se produce un cambio de punto de acceso por parte del MN, entonces, el LMA, tal y como se especifica en [5], deberá recibir dos PBUs correspondientes al mismo MN, uno de conexión y otro de desconexión, y éste únicamente deberá enviar un PBA al nodo que le envió el PBU de conexión. La siguiente figura muestra estos mensajes recibidos en el LMA.



No.	Time	Source	Destination	Protocol	Info
1	0.000000	2001::3	ff02::1:ff00:1	ICMPv6	Neighbor solicitation
2	0.000026	2001::1	2001::3	ICMPv6	Neighbor advertisement
3	0.000101	2001::3	2001::1	MIPv6	Binding Update
4	0.000417	2001::1	2001::3	MIPv6	Binding Acknowledgement
5	4.355664	2001::3	2001::1	MIPv6	Binding Update
6	4.999564	fe80::227:19ff:feb0:3	2001::3	ICMPv6	Neighbor solicitation
7	4.999655	2001::3	fe80::227:19ff:feb0:3	ICMPv6	Neighbor advertisement
8	7.795069	2001::2	ff02::1:ff00:1	ICMPv6	Neighbor solicitation
9	7.795099	2001::1	2001::2	ICMPv6	Neighbor advertisement
10	7.795175	2001::2	2001::1	MIPv6	Binding Update
11	9.795539	2001::1	2001::2	MIPv6	Binding Acknowledgement
12	9.997466	fe80::227:19ff:feb0:3	fe80::227:19ff:feb0:3	ICMPv6	Neighbor solicitation
13	9.997487	fe80::227:19ff:feb0:3	fe80::227:19ff:feb0:3	ICMPv6	Neighbor advertisement

Figura 24: Secuencia de mensajes LMA cuando MN cambia de *router*

En esta figura se muestra como inicialmente el MN se conecta a la red del MAG 1, éste le envía un PBU al LMA, y es asentido mediante un PBA (como se ha explicado en las anteriores figuras de esta misma sección). Pasado un cierto tiempo, el LMA vuelve a recibir un PBU del mismo MAG, en este caso un PBU de desconexión, mostrado en la *fig.* 26. A continuación se recibe un PBU de conexión del MAG-2, su contenido está mostrado en la *fig.* 22, y en este caso sí es asentido con un PBA, ya que contendrá la misma información que el mostrado en la *fig.* 23.

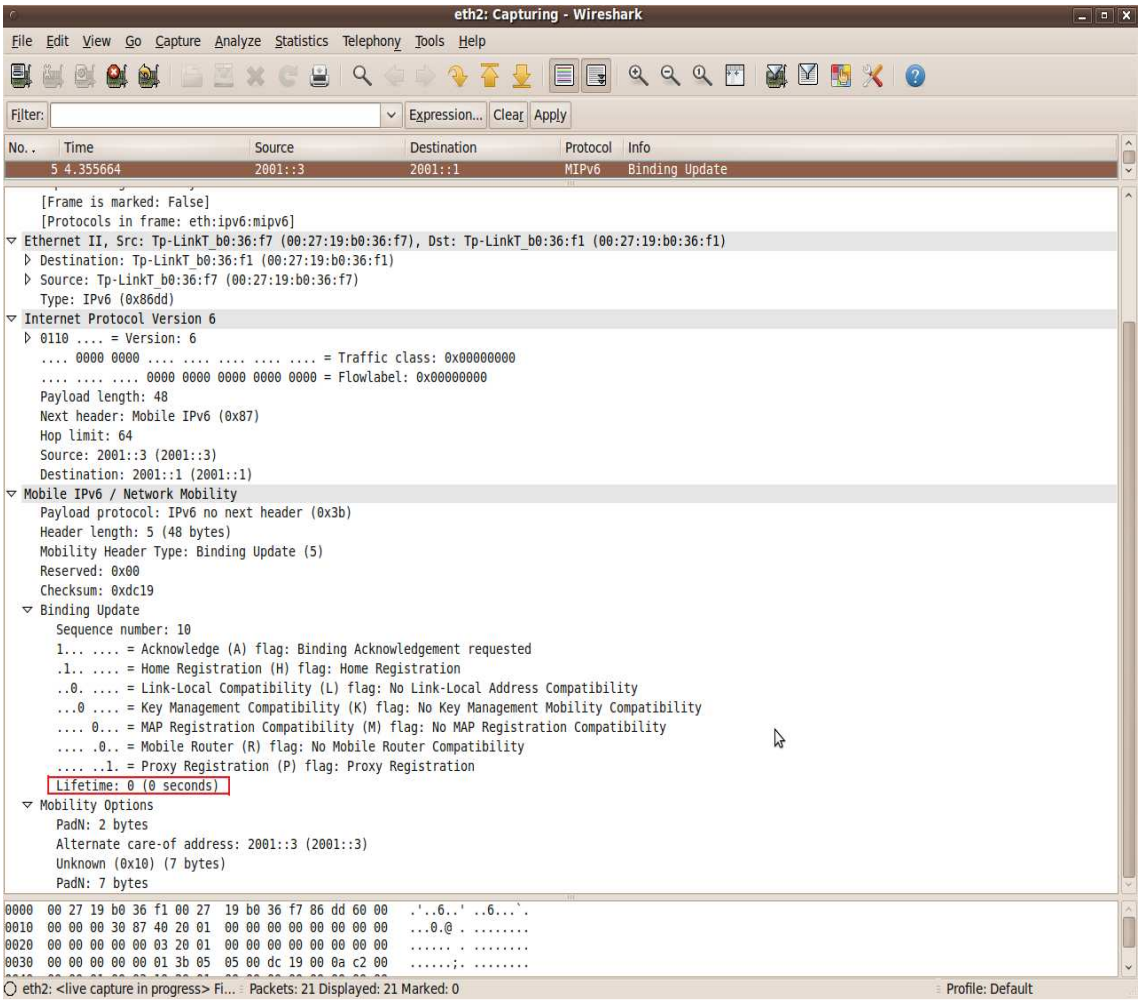


Figura 25: Mensaje PBU de desconexión MAG-LMA

Se puede contemplar que en este caso, el PBU es de desconexión debido a que su tiempo de vida es de 0, y corresponde con el mismo MN del que se recibió un previo PBU de conexión, dándole de esta forma de alta, y asignándole el prefijo “2002::”.

No se muestra el RA que envía el MAG-2 al MN debido a que su contenido es el mismo que el de la *fig.* 24, ya que el prefijo que se le especifica coincide con el que se le asignó en el caso anterior. De esta manera, al volver a configurar una dirección IPv6 de alcance global, volverá a configurar la misma que anteriormente. Además, ya que se forzó que las interfaces que conectaban los MAGs con los *routres* tuvieran la misma dirección IPv6 de ámbito global, la dirección origen del paquete también será la misma.

Una vez realizado esto se comprobará la tabla de rutas IPv6 existente en cada ordenador, así como la existencia del túnel que debe comunicar al MAG con el LMA. Esto es necesario para que se pueda producir una comunicación entre el MN y el LMA.

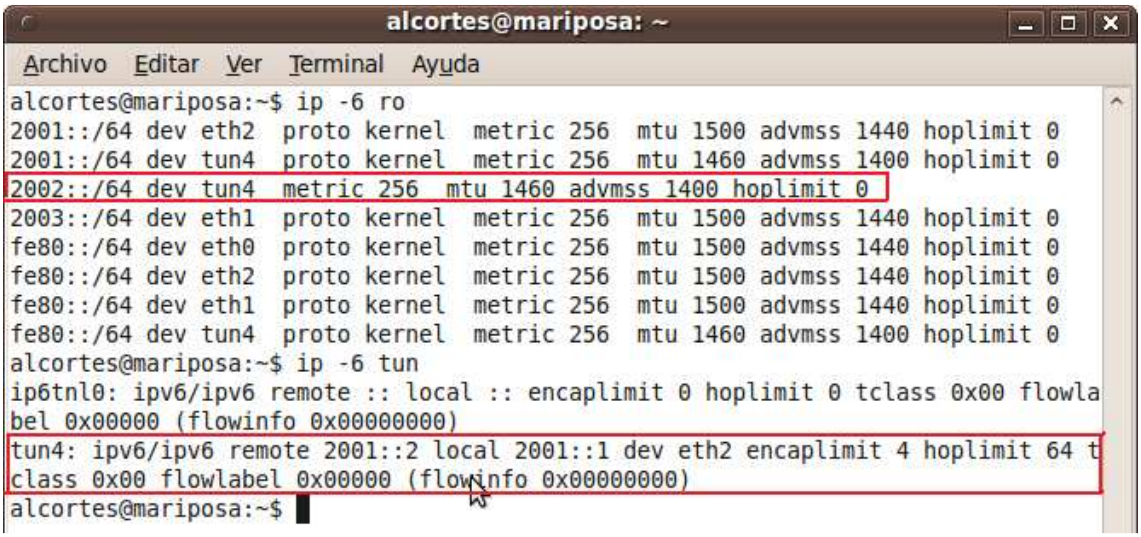


Figura 26: Rutas y túnel en LMA

Esta figura representa las rutas y túneles de IPv6 pertenecientes al LMA una vez que el programa desarrollado ha sido ejecutado, y antes de que el MN abandone la red del MAG (en tal caso tanto las rutas como el túnel serán eliminados por el mismo programa). En ella se puede comprobar la existencia de un túnel que comienza en el LMA de nuestro entorno de pruebas y cuyo final es el MAG-2. Además existe una ruta, que encamina todos los paquetes destinados a las direcciones que comiencen por el prefijo asignado al MN, dado que de primeras no se conocerá la dirección que éste configurará, por el túnel de comunicación creado entre el LMA y el MAG.

La siguiente figura, corresponderá con las rutas, túneles y vecinos que han sido creados por el programa en el MAG-2, para que pueda existir la comunicación deseada.

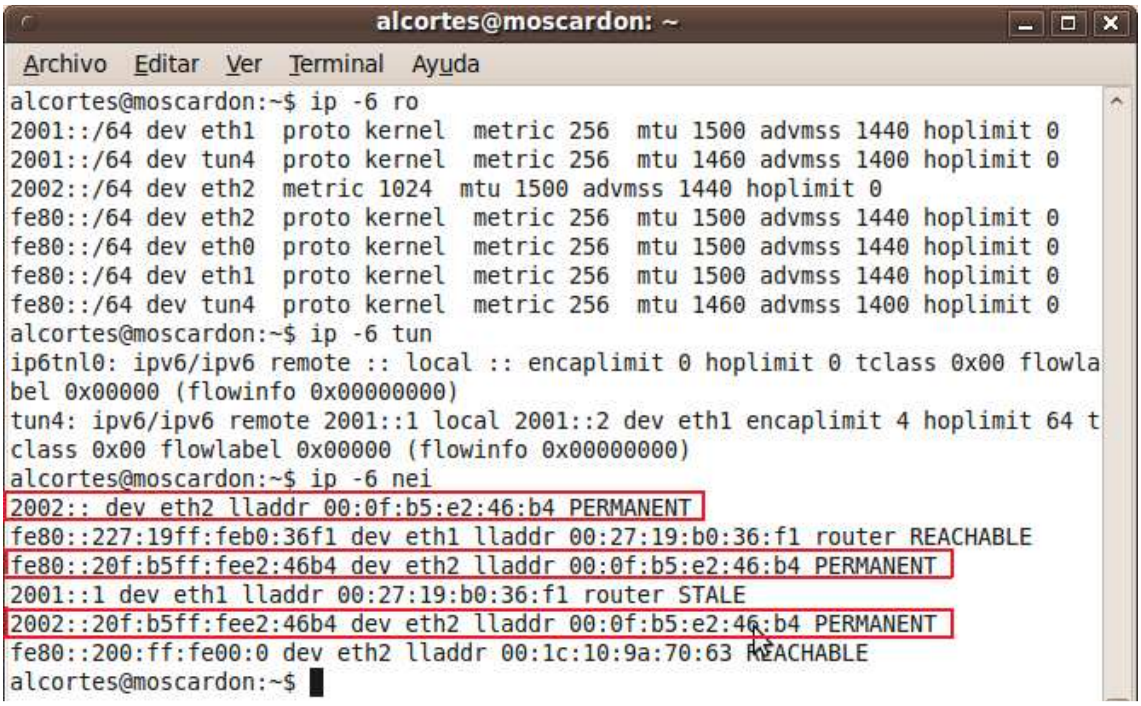


Figura 27: Rutas, túnel y vecinos en MAG

Al igual que en el LMA, se crea un túnel, y las rutas necesarias para comunicar a todos los elementos del entorno de pruebas. Pero lo más interesante de esta imagen, es la creación del MN como vecino, dado que de no hacerlo, la comunicación podría verse interrumpida en algunos casos. Al forzar a que el MAG y el MN sean vecinos, se incrementa la robustez del programa, ya que es una manera de asegurar que el MN es accesible por otros nodos.

A continuación se muestra una captura del programa *Wireshark* correspondiente con la interfaz por la que un MAG está conectado al punto de acceso al que se conecta el nodo móvil. En este caso se muestra la secuencia de mensajes, incluidos los *echo request* y *echo reply*. La comunicación entre el LMA y el MN únicamente será posible una vez que se hayan llevado a cabo todas las acciones anteriores.

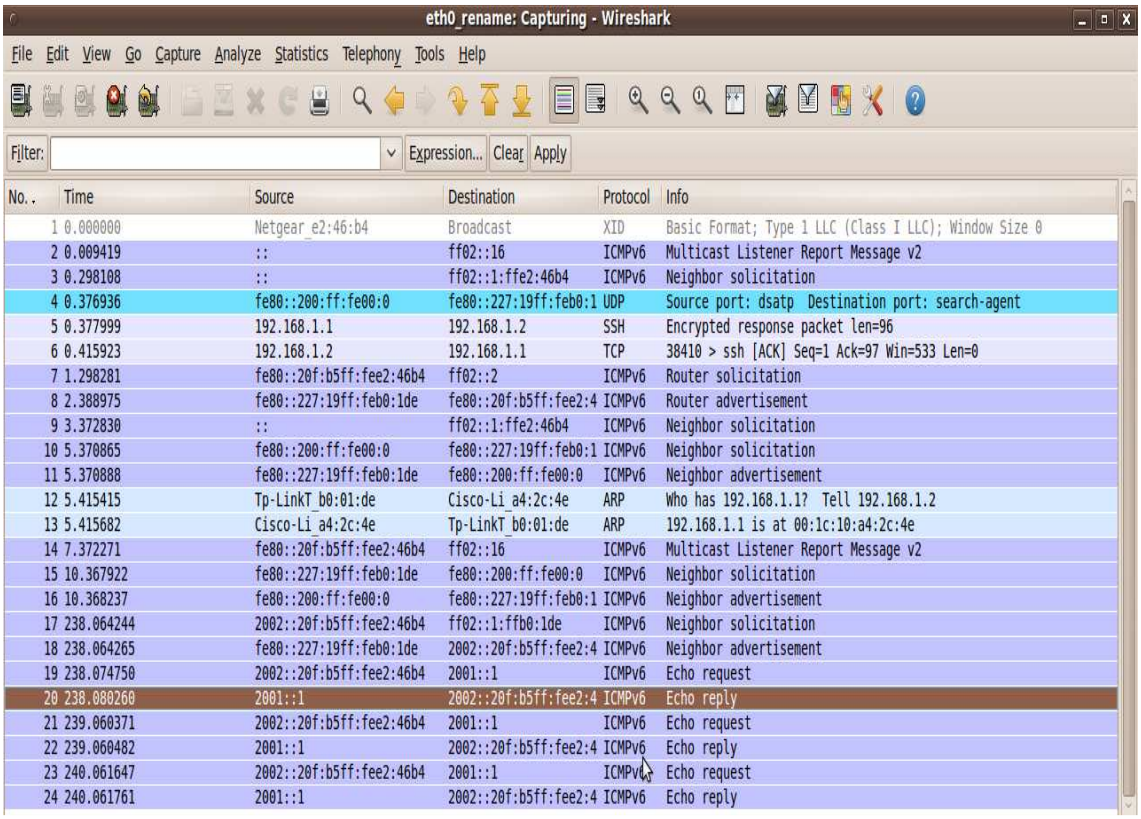


Figura 28: Secuencia mensajes *ping6* en MAG

En ella, se observa como el MN envía el mensaje de tipo *echo request* y el LMA le contesta mediante un *echo reply*.

5. 5 Evaluación del tiempo

Una vez comprobado el correcto funcionamiento de la implementación realizada, deberá ser evaluado su comportamiento. El factor más importante a evaluar, es el tiempo en el que un MN es inaccesible por otros nodos de la red una vez se haya desplazado de ubicación, es decir haya cambiado de punto de acceso a la red.

Para obtener una conclusión sobre el tiempo de incomunicación, se propone un análisis estadístico, en el cual, mediante la toma de una serie de datos, se pueda concluir el tiempo en el que el MN, en el entorno de pruebas propuesto, no sea accesible. Evidentemente, a menor tiempo necesitado para restablecer la comunicación, más eficiente será el programa.

Lo primero a realizar, para llevar a cabo este análisis, será la toma de datos. En este caso, el tamaño de la muestra, es decir de la población será de 30 datos. Cada uno de los datos corresponde con el tiempo de incomunicación del MN tras un cambio de punto de acceso a la red. Los datos son tomados en una ejecución normal del programa, es decir, sin forzar ninguna

situación extraordinaria, ya que se ha comprobado que la implementación funciona adecuadamente, y es estable. El proceso a realizar, será forzar al MN a conectarse cada vez a un *router* distinto, que a su vez está conectado a un MAG. Así, por ejemplo, cuando esté conectado al MAG-1, hacer que se conecte al MAG-2, y viceversa. Cada una de estas operaciones significará un cambio de punto de acceso a la red, por lo que se deberá realizar un total de 30 cambios. Los comandos necesarios para realizar estos cambios han sido explicados en la sección *Validación del funcionamiento*, perteneciente a este mismo capítulo.

En cuanto a la toma de datos, se utilizará el comando *ping6*, perteneciente al protocolo ICMPv6. A este comando, se le puede pasar como parámetro el intervalo de tiempo que se desea que exista entre que se envía un *echo request*, y se envía el siguiente. Cuanto menor sea este intervalo, más precisión existirá en las medidas de tiempo que se recojan. El tiempo escogido para ese intervalo corresponde con 50 ms. Destacar, que para poder ejecutar este comando con dicho intervalo de tiempo, debe ser ejecutado con permisos de *root*, dado que el intervalo más pequeño que un usuario sin dichos privilegios puede ejecutar es de 0.2 s. Lo que es mostrado por pantalla por este comando, se redirigirá a un archivo de texto, denominado "medidas.txt", para posteriormente poder ser analizado. El comando ejecutado, por tanto, es:

#ping6 2001::1 –i 0.05 >medidas.txt 2>l1

Dicho comando es ejecutado en el MN, e irá mandando paquetes *echo request* cada 50ms al LMA (2001::1). En el fichero "medidas.txt", se guardarán todas las respuestas recibidas, *echo reply*, que envía el LMA al MN. En él, se podrá observar los datos obtenidos. Este comando se lanzará cada vez que se vaya a realizar un cambio de punto de acceso, por lo que el fichero "medidas.txt" se sobrescribirá. Lo interesante será mirar los números de secuencia de las respuestas obtenidas, y cuando no sean números sucesivos, implicará una pérdida de paquetes, debido a que el MN ha cambiado el punto de acceso. Por lo tanto, se podrá saber el número de paquetes que no han podido ser recibidos, debido a que el nodo no era accesible. Dado que se conoce el intervalo de tiempo en el que se debería recibir cada paquete, 50ms, el tiempo total de incomunicación en cada caso corresponderá a realizar una multiplicación del número de paquetes perdidos por el tiempo en que se debe recibir cada paquete. Por tanto, el tiempo perdido en cada cambio de punto de acceso se calculará mediante la siguiente expresión:

Tiempo incomunicación = nº paquetes perdidos * tiempo de envío de cada paquete

El tiempo de envío de cada paquete siempre será el mismo y corresponderá con 50ms. El tiempo obtenido tras este cálculo será, el menor tiempo posible en el que el MN haya estado incomunicado, ya que al medir este tiempo de esta forma, hay que tener en cuenta que puede ser mayor del calculado, en concreto 50 ms más, debido a que es la diferencia de tiempo que existe entre dos paquetes, por lo que la comunicación se restableció entre el último paquete perdido y el primer paquete recibido después del cambio de punto de acceso a la red. Entonces, el tiempo de incomunicación exacto se encontrará en el intervalo del tiempo

calculado y 50 ms más. Así, los datos obtenidos tras la realización de la simulación anteriormente descrita, están representados en la siguiente tabla:

	Nº secuencia inicial	Nº secuencia final	Nº paquetes perdidos	Tiempo total (ms)
1	110	142	31	1550
2	104	128	23	1150
3	88	103	14	700
4	413	439	25	1250
5	77	84	6	300
6	80	90	9	450
7	927	948	20	1000
8	63	71	7	350
9	588	604	17	850
10	61	94	32	1600
11	74	86	11	550
12	61	77	15	750
13	106	126	19	950
14	74	104	29	1450
15	60	80	19	950
16	77	113	35	1750
17	74	87	12	600
18	123	129	5	250
19	77	102	24	1200
20	69	98	28	1400
21	84	96	11	550
22	60	83	22	1100
23	64	82	17	850
24	71	108	36	1800
25	66	95	28	1400
26	57	66	8	400
27	54	68	13	650
28	80	86	5	250
29	60	82	21	1050
30	69	76	6	300

Tabla 2: Datos del tiempo de incomunicación

El número de secuencia inicial y final, muestran el primer número de secuencia dónde este campo deja de tener valores sucesivos, y el número final muestra el número de secuencia del siguiente paquete recibido.

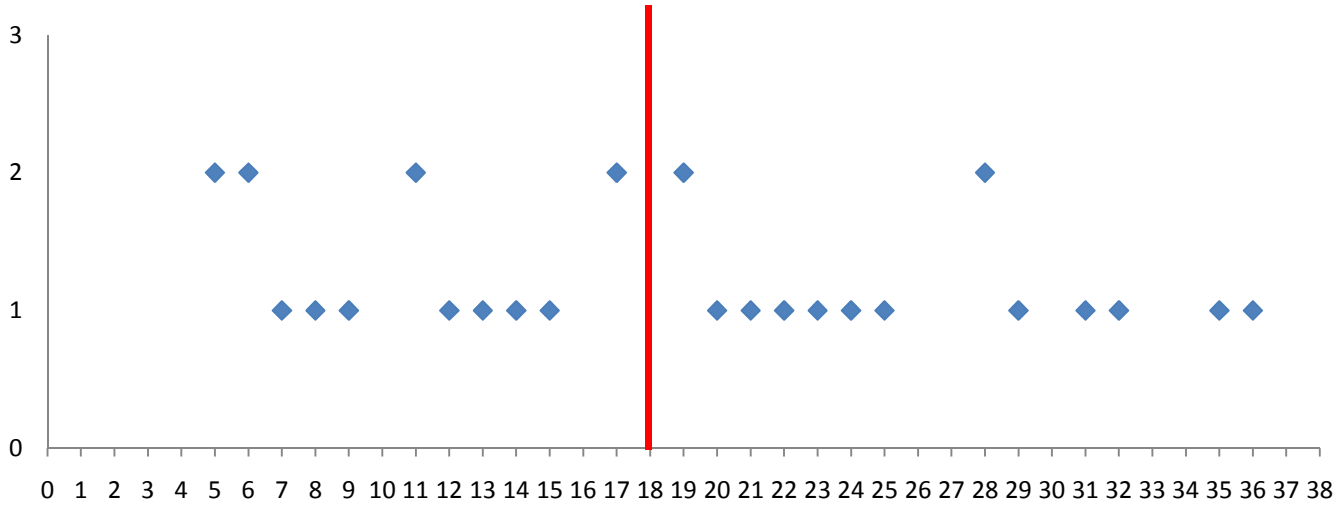
A simple vista, se puede destacar que:

Mínimo: 5 paquetes. 250 ms.
Máximo: 36 paquetes. 1800 ms.

Para comenzar con el análisis estadístico, el primer paso a realizar será conocer el tipo de distribución por la cual se distribuyen los datos. Para esto, se utilizarán el número de mensajes perdidos en cada uno de los cambios de acceso, debido a que el rango de valores es

menor. A la hora de calcular el intervalo de confianza, se realizará con el tiempo de incomunicación en microsegundos.

Lo primero que se realizará para conocer la distribución de los datos, será un grafico de dispersión de los mismos. En el eje X se representará el número de paquetes perdidos, y en el eje Y el número de repeticiones de ese valor.



Gráfica 1: Gráfico de dispersión de datos de incomunicación

Tras las 30 repeticiones, fácilmente, se puede comprobar que la mediana de los datos corresponderá con 16, dado que tanto por encima como por debajo de él se encuentran 15 datos.

A continuación, se procede a calcular la media, mediante la siguiente ecuación:

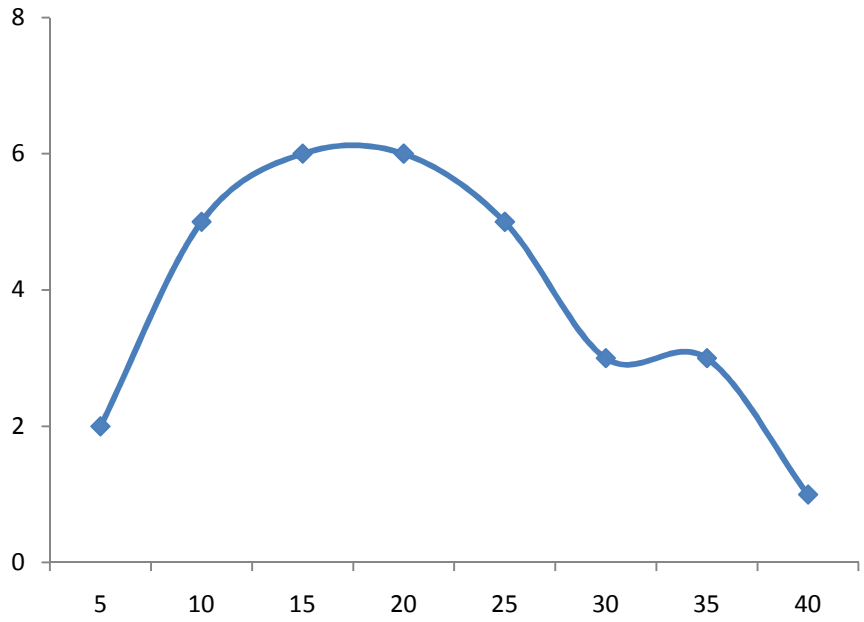
$$\bar{x} = \frac{\sum_i^{30} x_i}{30} = \frac{548}{30} = 18{,}26 \approx 19\ paquetes$$

Se aproxima al entero mayor, dado que no se puede perder una fracción del paquete, en caso de perderse, se perderá entero.

Por tanto, la media y la mediana están muy próximas. Además, se puede observar en el gráfico de dispersión, que los datos se encuentran muy concentrados entorno a la media, por lo que se puede deducir que los datos se distribuirán según una **NORMAL** [17].

Si ahora se dividen los datos en ocho diferentes grupos (hasta 5, de 5 a 10, de 10 a 15, de 15 a 20, de 20 a 25, de 25 a 30, de 30 a 35 y mayores de 35), dependiendo del número de paquetes que se hayan perdido, sumando el número de datos correspondiente a cada

intervalo, al representarlo, se obtiene la siguiente gráfica, dónde se puede observar una cierta similitud con una distribución Normal:



Gráfica 2: Gráfica de distribución de datos

Según se incrementen el número de muestras tomadas, la gráfica se parecerá más a la gráfica de una distribución Normal. Esto es debido a que la muestra con la que se está trabajando es pequeña. En estos casos, en lugar de trabajar con la distribución Normal, se estudiarán los datos distribuidos en una *t-STUDENT*.

El siguiente paso en el estudio estadistico, consistirá en calcular un intervalo de confianza. Este concepto puede ser definido como aplicar una regla para especificar los extremos de un intervalo, dónde, con una determinada probabilidad, se encontrará un parámetro verdadero. La probabilidad que va a ser aplicada en este caso será del **95%**.

Dado que la población, es decir el número de muestras tomadas, es pequeña, ya que no supera las 30, se utilizará la distribución *t-STUDENT*. Esta distribución, es una distribución de probabilidad dónde las muestras estan distribuidas en una normal, y el tamaño de la población es pequeño. Además las variables deben ser aleatorias e independientes, como es el caso.

Por tanto, el estimador utilizado en este caso será:

$$\bar{x} \sim \mathrm{N}\left(\mu, \frac{\vartheta}{\sqrt{n}}\right)$$

El estadístico utilizado, se obtendrá tipificando la distribución de $\bar{x}$, mediante:

$$\frac{\bar{x} - \mu}{\hat{s}/\sqrt{n}}$$

Dónde:

$$\widehat{S} = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$

El intervalo de confianza, vendrá dado, por:

$$(\bar{x} - L, \bar{x} + L)$$

Dónde L será igual a:

$$L = t_{n-1;\alpha/2} \frac{\hat{S}}{\sqrt{n}}$$

α corresponderá con el nivel de significación del intervalo de confianza. Por tanto:

$1 - \alpha = 0.95 \longrightarrow \alpha = 0.05$

n es igual al número de muestras tomadas, por lo que: **n=30**.

$t_{n-1;\alpha/2}$ será el valor correspondiente de la table t- STUDENT (Anexo H) para una población n-1 del nivel de significación, en este caso 0.025. Por lo que valdrá **2.045**.

De la misma manera que se realizó anteriormente con el número de paquetes perdidos, ahora será necesario calcular la media con el tiempo en el que el MN no puede enviar ni recibir ningún paquete, puesto que el intervalo de confianza se realizará con esa variable:

$$\bar{x} = \frac{\sum_{i}^{30} x_i}{30} = \frac{27400}{30} = 913,33 \approx 950\, ms$$

El siguiente paso será calcular el valor de la cuasi-desviación típica, $\hat{S}$:

$$\widehat{S} = \sqrt{\frac{6020000}{29}} = \sqrt{207586,207} = 455,61 \approx 500\, ms$$

Por último, se calculará L:

$$L = t_{29;0.025} * \frac{500}{\sqrt{30}} = 2{,}045 * 91{,}2870 = 186{,}68 \approx 200 \text{ ms}$$

Por tanto, se podrá asegurar que con una probabilidad del 95%, el tiempo de incomunicación del MN, se encontrará en el intervalo de: **(750, 1150)ms**.

Todos los datos han sido redondeados al múltiplo de 50 inmediatamente superior al valor obtenido, dado que es con esa precisión con la que se tomaron los datos de muestra.

5. 6 Estudio del tiempo de incomunicación

Después de la obtención de los resultados del análisis estadístico es conveniente conocer el porqué de ese tiempo de incomunicación. Cuando un MN cambia de punto de acceso a la red, se deben realizar diversas tareas, y cada una de ellas consume un tiempo, y la suma total de esos tiempos será el tiempo de incomunicación calculado anteriormente.

Las tareas que deben ser realizadas para que el MN vuelva a ser accesible por los demás nodos de la red, tras un cambio de ubicación son:

1. El MAG al que se conecta debe darse cuenta de que un nuevo MN se ha conectado a su red. En esta operación, influirá el tiempo que necesite el MN para conectarse a la red, y el tiempo de envío del mensaje por parte del punto de acceso al MAG informando de dicho evento.
2. Una vez recibido el evento, el MAG deberá crear un PBU informando al LMA del mismo.
3. El LMA recibe el PBU, gestiona los datos y envía un PBA de asentimiento.
4. El MAG recibe el PBA, gestiona los datos y envía un RA al MN.
5. El MN recibe el RA, y se auto configura. Entonces, ya vuelve a ser accesible por los demás nodos de la red.

Conviene mencionar que todas las tareas son secuenciales, es decir, una tarea no puede comenzar hasta que su predecesora haya terminado. Por ejemplo, el LMA, no podrá empezar a gestionar los datos de un MN hasta que no haya recibido el PBU del MAG, además, no conocerá qué MN, ni la información relativa a él, se ha conectado a la red hasta recibirlo.

El tiempo de auto configuración es despreciable, dado que en realidad, para el MN su *router* por defecto siempre es el mismo, puesto que ambos tienen la misma dirección IP, y excepto la primera vez que se conecta a la red, al recibir el RA, éste únicamente refrescará la entrada con el tiempo de vida del prefijo, puesto que el prefijo coincidirá, también, con el que se le asignó anteriormente, y por lo tanto la dirección IPv6 relativa a ese prefijo ya habrá sido creada previamente.

Por tanto, se podría hacer una división de tiempos, entre las diferentes tareas. La nomenclatura empleada corresponde con:

- **T_{L2}:** lo primero que debe realizar el MN es conectarse a un nuevo punto de acceso, para que esta acción sea llevada a cabo es necesaria un intercambio de mensajes entre el MN y el correspondiente punto de acceso. Esta variable medirá el tiempo que consume dicha señalización.
- **T_{MAG-NE}:** este tiempo corresponderá con el tiempo que necesite el punto de acceso para enviar el mensaje del evento de tipo 1, es decir de conexión, al MAG correspondiente, y éste último lo reciba, una vez que se haya detectado que un nuevo MN se ha conectado a su red.
- **$T_{PROC-NE}$**: este tiempo corresponderá con el necesario para que el MAG procese en nuevo evento recibido y envíe el PBU correspondiente.
- **$T_{MAG-LMA}$**: tiempo que tarda el PBU en alcanzar su destino. Es decir desde que lo envía el MAG hasta que el LMA lo recibe. Este tiempo, y el tiempo que tarde el PBA desde el LMA al MAG se considerará el mismo, dado que será el mismo camino el que ambos paquetes deban recorrer, pero en sentido opuesto.
- **$T_{PROC-PBU}$**: será el tiempo de gestión de datos en el LMA. Desde que recibe el PBU hasta que envía el PBA.
- **$T_{PROC-PBA}$:** corresponderá con el tiempo en que el MAG está gestionando los datos del PBA recibido. Es decir, la diferencia de tiempo, entre que recibe el PBA y envía el RA al MN.
- **T_{MAG-MN}**: corresponderá con el tiempo de envío del RA desde el MAG al MN. Este tiempo no se ha tenido en cuenta a la hora de realizar el estudio del tiempo dado que únicamente afectaría cuando el MN se une por primera vez a la red del entorno de pruebas montado, pero no cuando se produzcan cambios de punto de acceso de la misma red. Esto es debido por lo que se ha comentado de la auto configuración del nodo.

Así:

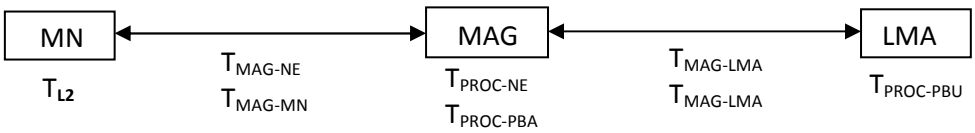


Figura 29: Distribución de tiempos

El tiempo total, calculado en la sección anterior, correspondería con la siguiente expresión:

$$T= T_{L2}+ T_{MAG-NE} + T_{PROC-NE} + 2* T_{MAG-LMA} + T_{PROC-PBU} + T_{PROC-PBA}.$$

$T_{MAG-LMA}$ es multiplicado por dos, porque la acción que representa sucederá dos veces, una cuando el MAG envíe el PBU al LMA, y otra cuando éste último le conteste mediante un PBA.

No es necesario que los relojes de todos los equipos pertenecientes al entorno de pruebas estén sincronizados, para conocer todas las componentes del tiempo total de incomunicación.

A la hora de conocer T_{L2}, será necesario poner la tarjeta de red del MN con la cual se conecta a los diferentes puntos de acceso en modo monitor, para poder, de esta manera conocer todo el tráfico *wireless* que esta interfaz recibe o envía. Al realizarlo como describe [18], se crea una nueva interfaz virtual asociada a la tarjeta de red inalámbrica que se desee. Una vez creada, y gracias al programa *Wireshark*, se capturará todo el tráfico *wireless*. Desde que se inicia el proceso de conexión a una red inalámbrica, hasta que finalmente un nodo se conecta a ella, son necesarios los siguientes mensajes [19]:

- ***Deauthentication***: mensaje que envía un nodo a un punto de acceso cuando el primero se encuentra conectado al segundo, para indicarle que va a abandonar la red.
- ***Probe request:*** mensaje que envía un nodo con destino *broadcast* para preguntar por el punto de acceso correspondiente a la red a la que dicho nodo se quiere conectar.
- ***Probe response:*** mensaje enviado por el punto de acceso al nodo que envío el *probe request*, para comunicarle que es él al que debe conectarse.
- ***Authentication:*** serán necesarios dos mensajes de este tipo, uno desde el nodo hacia el punto de acceso y el otro en sentido contrario. Ambos, como su propio nombre indica servirá para autenticar ambos nodos.
- ***Association request:*** una vez autenticados, se comienza la conexión al punto de acceso seleccionado. Para ello, el nodo envía una petición de conexión a dicho punto de acceso.
- ***Association response:*** contestación del punto de acceso al nodo que le envió el *association request*. Una vez que se recibe este mensaje, el nodo ya estará conectado a la red del punto de acceso.

Por tanto, para calcular T_{L2}, será necesario capturar todos los paquetes comentados, y el tiempo de retardo que existirá entre que se inicia la acción de conectarse hasta que el MN está realmente conectado a la red, será el tiempo que haya pasado desde que se envió el primer mensaje explicado hasta que se recibió el último. Por tanto, será resultado de restar el tiempo de recepción del mensaje *association response* y el tiempo de envío del mensaje de *deauthentication*.

T_{MAG-NE} no será calculado durante la misma ejecución en la que se obtienen los demás datos, puesto que para que su cálculo sea más sencillo, y se pueda observar los datos necesarios en un mismo ordenador, la dirección a la que el programa que se ejecuta en el *router* manda el mensaje UDP, no será la correspondiente con el MAG al que se encuentra conectado si no la del MN. Por lo tanto, será necesario conectar el MN al *router* mediante un cable. En este ordenador, con la tarjeta de red en modo monitor, se podrá conocer cuando se recibe el mensaje *association response*, y cuando se recibe el mensaje UDP originado en el

router. La diferencia de tiempo de ambos mensajes será el tiempo que necesita el *router* para generar y enviar el mensaje al MAG. Como en los demás casos, la hora de recepción de los mensajes se conocerá gracias al programa *Wireshark.*

En cuanto $T_{PROC-NE}$, será calculado, utilizando *Wireshark*, el cual nos indicará cuando el MAG recibió el mensaje del punto de acceso indicando que un nuevo MN se ha conectado a su red, y cuando envió el PBU al LMA. La diferencia de ambos, será el valor de este dato.

Para calcular $T_{PROC-PBU}$ también se hará uso del *Wireshark*, puesto que la diferencia de tiempos entre que el LMA manda el PBA y recibió el PBU, será el tiempo que ha necesitado para gestionar el PBU.

Calcular $T_{MAG-LMA}$, es un poco más complicado, pues serán necesarias realizar más operaciones para obtener este dato. Mediante la aplicación *Wireshark*, se puede conocer el tiempo transcurrido desde que el MAG envió el PBU hasta que recibió el PBA. Llamemos a este tiempo Tr. $T_{PROC-PBU}$ indica el tiempo que el LMA necesita para administrar los datos recibidos, y realizar con ellos las operaciones necesarias. Pues bien, si se resta Tr – $T_{PROC-PBU}$, se obtendrá 2* $T_{MAG-LMA}$, que justamente es lo necesario para calcular el tiempo total, por lo que no será necesario realizar la división entre 2 para calcular el dato exacto. También podría ser obtenido calculando el RTT entre el MAG y el LMA.

Como en los casos anteriores, gracias al *Wireshark*, se podrá calcular $T_{PROC-PBA}$, dado que será la diferencia entre que el MAG recibe el PBA y envía el RA al MN.

Al igual que en el apartado anterior, se han tomado 30 muestras de los diferentes datos necesarios (cada una de las muestras pertenece a la misma ejecución del programa que en el apartado anterior, exceptuando las relativas a T_{MAG-NE}), para poder realizar el mismo análisis que anteriormente, y poder estimar de una manera más exacta la distribución del tiempo en el que un MN no puede ni enviar ni recibir paquetes.

Resumiendo, los datos y las operaciones necesarias para el cálculo de los diferentes tiempos serán:

- Para calcular T_{L2}, será necesario conocer la hora en la que el MN comenzó la desconexión del anterior punto de acceso (Tda), y la hora en la que recibió la confirmación del nuevo punto de acceso como que está ya conectado a él (Tas). **T_{L2}= Tas-Tda.**
- A la hora de calcular T_{MAG-NE} será necesario conocer la hora de envío del mensaje *association response* que indicará que el punto de acceso es consciente de que un nuevo MN se ha conectado a él (Tas), y la hora en la que el mensaje que el *router* envía informando de tal evento al MAG (Tne). La resta del segundo menos el primero dará el valor deseado. Así: **T_{MAG-NE} =Tne – Tas**.
- Para calcular $T_{PROC-NE}$ se utilizan el tiempo de llegada del mensaje del *router* al MAG (Tm) y hora de envío del PBU del MAG hacia el LMA (Tspbu). **$T_{PROC-NE}$ =Tspbu-Tm.**

- Para calcular $T_{PROC-PBU}$ será necesarios: hora de recepción del PBU (Trpbu) y la hora de envío del PBA (Tspba). **$T_{PROC-PBU}$ =Tspba-Trpbu**.
- Una vez obtenidos los datos anteriores, 2* $T_{MAG-LMA}$ será preciso conocer la hora a la cual se envió el PBU (Tspbu), la hora a la que se recibió el PBA (Trpba), y $T_{PROC-PBU}$. **2* $T_{MAG-LMA}$ =(Trpba-Tspbu)- $T_{PROC-PBU}$.**
- Cuando se quiera calcular $T_{PROC-PBA}$ se necesitará la hora de llegada del PBA (Trpba), y la hora de envío del RA (Tsra). **$T_{PROC-PBA}$ =Tsra-Trpba**.
- El RTT se calculará gracias a la hora de envío del *echo request* (Ter) y la hora de llegada del *echo reply* (Tey). **RTT=Tey-Ter**.

Los datos necesarios, vendrán expresados únicamente por los segundos y milisegundos de la hora a la que se recibieron/enviaron los diferentes mensajes. Todo ellos corresponden con la misma ejecución del protocolo. Es decir, en cuanto se produce un cambio del punto de acceso del MN en la red del entorno de pruebas, en el MN se calculará el tiempo total que ha estado incomunicado, mediante el número de paquetes perdidos del comando *ping6*, además será necesario examinar la captura del *Wireshark* relativa a la tarjeta de red en modo monitor, para calcular el tiempo que tardó el MN en conectarse a la red. Se tomarán también los datos de los tiempos relativos a la señalización del protocolo en el MAG y en el LMA, que se produjeron a raíz de que el MN se conectase al nuevo MAG

Una vez realizados los cálculos anteriormente explicados, se obtiene que:

T_{L2} (s)	T_{MAG-NE} (s)	$T_{PROC-NE}$ (s)	2* $T_{MAG-LMA}$ (s)	$T_{PROC-PBU}$ (s)	$T_{PROC-PBA}$ (s)	RTT (s)
1,0372	0	0,0003	0,0001	0,0012	0,0133	0,0012
0,2557	0,001	0,0003	0,0001	0,0042	0,0137	0,0109
0,0710	0,6752	0,0004	0	0,0032	0,0079	0,0014
1,1558	0,4178	0,0002	0,0002	0,0041	0,0075	0,0089
0,0521	0,3973	0,0004	0	0,0023	0,0082	0,0043
0,0427	0,7424	0,0002	0,0001	0,0004	0,0080	0,0024
0,7326	0,5159	0,0002	0,0002	0,0036	0,0127	0,0028
0,0518	0,241	0,0002	0,0001	0,0047	0,0288	0,0041
0,1618	0	0,0002	0,0001	0,0030	0,0062	0,0011
1,0727	0,4694	0,0003	0,0001	0,0038	0,0089	0,0012
0,0489	0,1838	0,0003	0	0,0022	0,0077	0,0021
0,0436	0,8463	0,0003	0,0001	0,0015	0,0080	0,0012
0,0432	0,2261	0,0004	0,0011	0,0005	0,0084	0,0021
1,0559	0,5137	0,0003	0,0001	0,0037	0,0084	0,0081
0,0526	0,3245	0,0004	0,0001	0,0014	0,0085	0,0042
1,0587	0,5028	0,0002	0,0001	0,0011	0,0085	0,0009
0,0540	0,6882	0,0003	0,0002	0,0014	0,0143	0,0035
0,0777	0,3567	0,0004	0,0001	0,0019	0,0139	0,0017
1,0511	0,2691	0,0003	0	0,0019	0,0076	0,0026
1,0498	0,6375	0,0003	0,0001	0,0034	0,0080	0,0017
0,0787	0,5967	0,0004	0,0001	0,0022	0,0067	0,0019
1,0466	0,1637	0,0003	0,0001	0,0011	0,0078	0,0060
0,0619	0,4969	0,0003	0,0002	0,0009	0,0081	0,0085
1,0596	0,2579	0,0002	0,0001	0,0013	0,0076	0,0021

1,0330	0,5197	0,0003	0	0,0025	0,0080	0,0241
0,0460	0,3162	0,0003	0	0,0018	0,0082	0,0037
0,0539	0,3308	0,0003	0	0,0025	0,0083	0,0009
0,0530	0,5113	0,0004	0,0001	0,0049	0,0136	0,0028
1,0309	0,7762	0,0002	0,0001	0,0033	0,0082	0,0047
0,2398	0,4347	0,0004	0	0,0019	0,0090	0,0032

Tabla 3: Datos calculados de tiempos

Se puede observar, que en varios casos de estudio el tiempo $T_{MAG-LMA}$ es equivalente a 0. Esto no significa que realmente sea 0, si no que el tiempo de envío es menor a la precisión que se ha utilizado para realizar este estudio.

Se aprecia que los tiempos T_{MAG-NE} son muy diferentes entre sí, esto es debido a que en el programa que se ejecuta en el *router*, dada su capacidad limitada, se le fuerza a que esté inactivo durante un segundo. Este retardo podrá variar entre 0 (como en varios casos), y 1 segundo, dependiendo de cuánto le quede al programa de estar inactivo cuando el MN se conecte a dicho *router*. A él habrá que sumarle el retardo ocasionado por el *driver* del dispositivo al detectar la conexión o desconexión de un nodo a él, para obtener las medidas tomadas.

Los retardos de transmisión podrían haber sido despreciados dado su bajo valor. Era de esperar obtener valores tan pequeños porque, como se ha comentado con anterioridad, el entorno de pruebas es el caso más básico, en el que todos los elementos están directamente conectados.

Al igual que en el apartado anterior, se procederá a calcular la media de todos los tiempos, así como el error que se haya podido añadir, mediante el uso de intervalos de confianza de las mismas características del apartado anterior. La tabla 4 muestra un resumen de los resultados obtenidos.

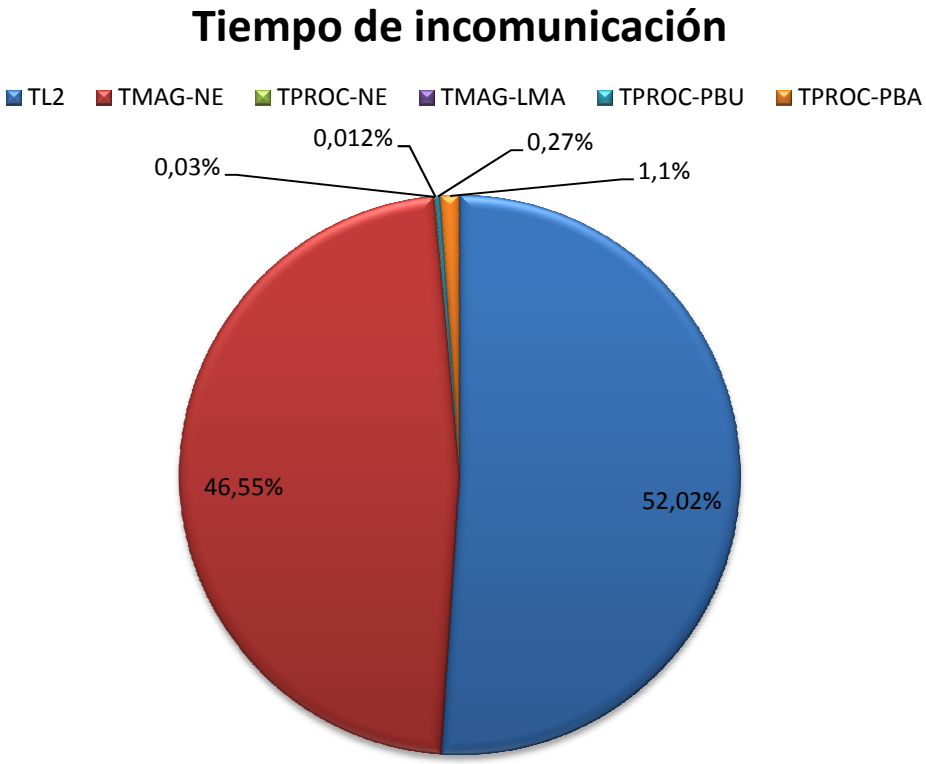
T_{L2} (s)	T_{MAG-NE} (s)	$T_{PROC-NE}$ (s)	2* $T_{MAG-LMA}$ (s)	$T_{PROC-PBU}$ (s)	$T_{PROC-PBA}$ (s)
0,4624 ± 0,1740	0,4137 ± 0,0739	$3 *10^{-4}$ ± $2,77*10^{-5}$	$1,2*10^{-4}$ ± $7,3*10^{-5}$	$2,4*10^{-3}$ ± $4,44*10^{-4}$	$9,8*10^{-3}$ ± $1,5*10^{-3}$

Tabla 4: Medias de tiempos

Una vez calculado el tiempo medio de cada uno de los tiempos que componen el tiempo de incomunicación, será interesante conocer el porcentaje de cada uno de ellos en el tiempo de incomunicación, para así conocer exactamente como dicho tiempo está distribuido. Por lo tanto:

- T_{L2}= 52,02 %
- T_{MAG-NE}= 46,55 %
- $T_{PROC-NE}$ = 0,03 %
- $T_{MAG-LMA}$ = 0,012 %
- $T_{PROC-PBU}$ = 0,27 %
- $T_{PROC-PBA}$ = 1,1 %

El siguiente gráfico representa la división del tiempo total, de acuerdo a los porcentajes calculados:



Gráfica 3: Distribución del tiempo de incomunicación

Se puede observar que la tarea que más tiempo consume es el tiempo que emplea el MN para la conexión a una red inalámbrica. Como se ha comentado anteriormente, este proceso requiere el intercambio de varios mensajes entre los elementos implicados, con lo cual, y tal y como se podría deducir, esta operación requiera un tiempo destacable dentro del total del tiempo de incomunicación.

A continuación, la siguiente tarea que emplea más tiempo, casi la mitad del total, corresponde con la creación y envío del mensaje UDP del *router* hacia el MAG. Esto es debido a que, como se ha comentado anteriormente, en el código del programa que lleva a cabo esta acción existe un retardo de hasta 1 segundo, por lo que depende del tiempo transcurrido desde que se inició ese retardo cuando el MN se conecte al punto de acceso del entorno de pruebas, lo que se tendrá que esperar para que se genere el mensaje a enviar, además del tiempo que emplee el dispositivo en detectar el evento producido.

El menor porcentaje, coincide con el tiempo que se necesita desde que un mensaje es enviado hasta que es recibido por el nodo destino. Esto, era de esperar, (y en algunos casos

dicha latencia es nula, según la precisión utilizada), dado que todos los nodos están directamente conectados, por lo que el tiempo de envío no será significante, e incluso podría haber sido despreciado del estudio.

El tiempo que el MAG emplea para enviar el PBU, una vez recibido el aviso del punto de acceso, es mucho menor que el tiempo que emplea para gestionar el PBA del LMA y enviar el RA. Esto es debido a que el MN se da de alta en la BUL, y se gestiona la demás información del mismo, una vez recibido el PBA, ya que antes no se sabrán todos los datos del mismo, por lo tanto, además de requerir algunas de las operaciones realizadas cuando se recibe el aviso del punto de acceso (cómo la búsqueda del perfil relativo a ese MN), será necesario realizar más operaciones.

El tiempo empleado por el LMA en realizar las operaciones necesarias, como asignar un prefijo al MN y darlo de alta en la *Binding cache*, pueden ser consideradas de la misma magnitud que las que realiza el MAG una vez recibido el PBA, pero el tiempo consumido es menor. Esto es debido a que el LMA dispone de más recursos HW que los MAG del entorno de pruebas, por lo que es razonable que éste pueda realizar las mismas operaciones en un tiempo menor. La diferencia tampoco es mucha por que la cantidad de operaciones que deben ser realizadas necesariamente no es muy grande.

Aunque en el tiempo total de incomunicación no se haya tenido en cuenta el RTT, resulta interesante calcular su tiempo medio, con su respectivo error, calculado de la misma manera que se calcularon el resto de tiempos en este proyecto.

La media obtenida del RTT es de: $4.1*10^{-3}$ s.

Ahora, al calcular el intervalo de confianza al 95%, obtenemos que, el RTT del entorno de pruebas propuesto en este caso, se encuentre dentro del intervalo ($2.5*10^{-3}$, $5.7*10^{-3}$) s con una certeza del 95%.

CAPÍTULO

6

CONCLUSIONES

Todo es muy difícil antes de ser sencillo.
Thomas Fuller (1610-1661)

6. 1 Conclusiones

Ante la creciente necesidad, en nuestra sociedad, de poder comunicarse así como de poder acceder a la información que se desee, en cualquier momento y en cualquier lugar, ha sido necesario el desarrollo de diversos dispositivos que lo permitan. Estos dispositivos tienen características dispares, dado que puede ser desde el tradicional ordenador de sobremesa con el que antes se accedía a Internet, como un teléfono móvil, que hoy en día, ya permite realizar casi las mismas operaciones que un ordenador. Las características de acceso a una red están cambiando, por lo que también han de cambiar los protocolos empleados para tal fin. Las redes de ordenadores ya no son redes de ordenadores fijos, puesto que no siempre se encontraran conectados a esa red los mismos nodos, ni en la misma posición. Dadas estas características de las redes, su estructura es variable, y puede ser modificada en cualquier momento. Debido a ello, será necesario que las redes se adapten a las nuevas necesidades, y puedan reestructurarse sin perder la comunicación entre los nodos que la forman.

Al existir más diversidad de dispositivos que pueden acceder a una red, y al ser muchos de ellos dispositivos móviles, es decir que pueden ir cambiando de posición mientras están conectados a una red, nace el concepto de movilidad IP. Tal y como se describió en el apartado 2.1, la movilidad puede ser considerada como la cualidad que posee un objeto para cambiar de ubicación. Es entonces cuando aparece el problema abordado a lo largo de este proyecto, conseguir que todos los nodos de una red permanezcan conectados a la misma aunque ellos cambien de ubicación, no sólo física, si no también dentro de la red. Además, no hay que descartar la posibilidad de que una red sufra una reestructuración debido al continuo movimiento de sus nodos, lo que conllevaría, posiblemente, un cambio de ubicación del nodo dentro de la misma.

Por ello, el nuevo protocolo IPv6 ya está mejor preparado para la posibilidad de movilidad, añadiendo una cabecera específica para tal efecto. En IPv4, también es posible abordar movilidad, pero el cómo se hace está fuera del alcance de este proyecto. Por tanto, este proyecto se ha centrado en IPv6, por lo que todos los protocolos descritos en él, resolverán el problema en el nivel de red de la torre TCP/IP, explicada en la sección 3.1.

Los problemas existentes cuando un nodo cambia de posición dentro de una red (provocado por cualquier motivo), podrían ser reducidos a que el nodo ha de generar una nueva dirección IPv6, y como consecuencia de ello, si está manteniendo una comunicación con otro nodo, éste le seguirá enviando los mensajes a su antigua dirección IPv6, con lo cual, la comunicación no sería satisfactoria porque dichos mensajes no serían recibidos, y lo que es más importante, puede darse la situación de que no se detecte que dichos mensajes se han perdido en la red.

El primer protocolo que es creado para resolver estos problemas, comentado en el apartado 3.1.2, y explicado detalladamente en [4], es conocido como MIPv6. Es en él, donde se definen varias entidades importantes como el HA. La solución que este protocolo propone consiste en que cada nodo tenga asignada siempre una misma dirección IPv6, nombrada como *home address*, mediante la cual, el HA podrá localizarle en cada momento. Cuando el nodo cambie de ubicación, deberá comunicarle su nueva dirección IPv6 al HA correspondiente, el cual mantendrá una asociación de la nueva dirección y de la *home address* del nodo. Todos los demás nodos que quieran comunicarse con el nodo, enviarán siempre los mensajes a la *home address* del nodo, por lo que será necesario que los reciba el HA, y sea éste el que los redirija hacia la nueva dirección. Con los mensajes enviados por el MN sucederá lo mismo, es decir, deberán pasar previamente por el HA antes de ser recibidos por el nodo destino.

A este protocolo se le asocian varios inconvenientes, dado que el MN será consciente de su cualidad de móvil, y por tanto de su movimiento, debiendo disponer de un SW específico para la comunicación con el HA, y además, y más importante, cambiará de IPv6 en cada cambio de ubicación. Es aquí donde se plantea una cuestión interesante, en analogía con los teléfonos móviles. Cuando un terminal abandona una antena que le da cobertura, y se la facilita otra, éste no cambia de número de teléfono, es decir de número de identificación, entonces, ¿por qué un nodo móvil debe cambiar de identificador cuándo cambia de punto de acceso a la red (la antena en el caso de los teléfonos móviles)?

Por ello, surge el protocolo PMIPv6 [5], objeto de estudio fundamental de este proyecto, cuyo funcionamiento es abordado en la sección 3.2. Sus ventajas principales sobre el protocolo MIPv6, son justamente, que no se encuentran los inconvenientes mencionados, es decir, el MN no dispone de ningún SW adicional, puede ser que no sea consciente de su movimiento, y además no cambiará de dirección IPv6 cada vez que cambie su ubicación. Esto es logrado gracias a que al aumento de los mensajes relativos a la señalización del movimiento, y a que, como su propio nombre indica, se añade una entidad intermedia, MAG, entre el HA y el MN que ayude a dicha gestión.

En el caso de PMIPv6, el HA será denominado LMA, y realizará básicamente las mismas funciones que su antecesor. El MAG, por tanto será el encargado de que el MN no sea consciente de su cambio de posición dentro de la red. Para ello, él actuará como su *router* por defecto, comunicándole el prefijo que le ha sido asignado dentro de la red. Este prefijo no variará a lo largo de la permanencia del MN en la red, por tanto, siempre que cambie de ubicación, el nuevo MAG al que se haya conectado, le indicará que es su *router* por defecto, y le asignará el mismo prefijo que ya poseía anteriormente. Entonces, el MN configurará una dirección IPv6 de acceso global que comience por dicho prefijo, y debido a que siempre se configuraran las direcciones usando el mismo algoritmo, siempre se configurará la misma, con el mismo prefijo. Al igual que en MIPv6, todos los paquetes con origen o destino al MN, deberán pasar previamente por el LMA para que sean procesados por el mismo.

Después de que este nuevo protocolo sea definido, el siguiente paso en la investigación será llevarlo a la práctica, para ello será precisa una implementación del mismo, y

comprobar que efectivamente, cumple con las expectativas previstas. Por ello, ciertas entidades, como la Universidad de Helsinki, desarrollan este protocolo.

Para poder examinar el funcionamiento de la implementación, es necesario instalarla en los equipos del entorno de pruebas. Para ello, previamente será necesario prepararlos para que cumplan ciertas características, y de esta forma puedan llevar a cabo determinadas funcionalidades del protocolo. Tanto dichas características, como la manera de realizarlo se encuentra descrita en el anexo G.

A lo largo del trabajo realizado con esta implementación, se encuentran dificultades, dado que el programa no tenía el funcionamiento esperado.

Como primera dificultad cabe destacar la ilegibilidad del código fuente, ya que al ser desarrollado por diversas entidades, cada una de ellas tiene una metodología de trabajar distinta, por lo que no existe uniformidad en el código. Además, aún se hace más difícil su entendimiento debido a que no existe documentación sobre ella, por lo que se tuvo que ir examinando el código detalladamente, para poder la manera de realizar el protocolo.

Pero sin lugar a dudas, el problema más significativo de esta implementación, es que no podrá ser ejecutada en un entorno real, dado que es necesario que siempre que se conecta un MN a un MAG, el primero le envíe un RS al segundo, y esto únicamente sucede si el MN activa en ese momento la interfaz inalámbrica por la que procede a conectarse a la red del MAG. Por ello, se decidió incrementar su funcionalidad, para que pudiese realizar las mismas operaciones del protocolo PMIPv6, sin necesidad de recibir ese mensaje, y de esta forma, adaptarlo a una ejecución más realística.

Tras los cambios necesarios en la implementación original, y tras incluir nuevas funciones para que este incremento sea llevado a la práctica se observan ciertas anomalías. Que el programa funcione correctamente, es totalmente aleatorio. Tras diversas comprobaciones, se puede asegurar que el problema no se encuentra en el código escrito por nosotros, dado que sucede lo mismo con la función que iniciaba la ejecución de las tareas de PMIPv6 de la implementación original.

Dado que el objetivo principal de este proyecto es conseguir una implementación que consiga realizar las funciones de PMIPv6 de una manera exhaustiva, y real, se decide, siguiendo esta idea, implementar una nueva para cumplir con dicho objetivo. Es entonces, cuando empieza una nueva fase en el desarrollo de este proyecto.

Debido a la gran extensión del protocolo PMIPv6, no se implementarán todas sus funcionalidades, únicamente las relativas a la señalización para registrar un nuevo MN, y las acciones necesarias para que éste sea accesible por los demás nodos de la red. Dicha comunicación debe ser restablecida siempre, cuando el MN cambie de punto de acceso, y a ser posible en el menor tiempo posible, para que el número de paquetes perdidos sea el mínimo.

Para un mayor detalle de las funcionalidades realizadas por esta versión reducida del protocolo PMIPv6 acudir al apartado 4.1, del capítulo trabajo realizado.

Una vez concluida la etapa de desarrollo del protocolo, éste ha de ser validado, para asegurar que se obtiene el resultado esperado. Dichas pruebas pueden ser observadas en el apartado 5.3. Queda por tanto demostrado que el objetivo inicial, ha sido cumplido, dado que a través del comando *ping6*, de conexiones SSH y de *streaming* de un video desde el LMA al MN utilizando el reproductor VLC, se demuestra que el MN y el LMA mantienen una comunicación satisfactoria mientras en MN está conectado a la red de un MAG, y tras verse interrumpida por un período de tiempo tras cambiar de posición en la red, dicha comunicación vuelve a ser restablecida. Además, se comprueba que la señalización enviada es la adecuada en cada momento. También ha de ser comprobado la creación de rutas y de túneles para que dicha comunicación pueda ser producida.

Al determinar que la implementación funcionaba adecuadamente, surgió una nueva duda ¿cuánto es el tiempo de incomunicación tras cambiar de punto de acceso a la red? En ciertas comunicaciones, o aplicaciones, el número de paquetes perdidos puede ser de gran importancia, como puede ser en los sistemas en tiempo real, lo que puede hacer que esta implementación no pueda ser utilizada dependiendo de lo crítico que resulte el tiempo en las aplicaciones ejecutadas en las capas superiores de la torre TCP/IP. Por tanto se consideró necesario realizar un estudio del tiempo que era necesario para restablecer la comunicación entre el MN y el LMA.

Todos los detalles de dicho estudio, se encuentran en el capítulo 5 (apartados 5.5 y 5.6). En él, se podrá ver que el tiempo de incomunicación con una certeza del 95% se encontrará dentro del intervalo (750, 1150)ms, y teniendo como media 950 ms. Esto es poco más de un segundo, lo cual se considera un tiempo aceptable en las circustancias que ha sido calculado. En la realidad habría que procurar disminuir ese tiempo al mínimo posible, dependiendo de la criticidad de las acciones que el MN esté realizando mientras está conectado a la red. Volviendo a la similitud con el teléfono móvil, cuando éste cambia de antena, el tiempo que transcurre sin conexión es prácticamente despreciable, por lo que en el caso de PMIPv6 debería tratar de conseguirse lo mismo.

Cuando dicho tiempo había sido cuantificado, y dado que dependerá mucho del entorno de pruebas, o del entorno real en el que esta implementación sea ejecutada, se decidió realizar otro estudio similar al anterior, pero para conocer los componentes del tiempo de incomunicación, es decir, cuáles son las tareas que se realizan durante dicho período, y cuanto tiempo consume cada una de ellas. Con ello, se consigue estimar el porcentaje del tiempo total que necesita cada tarea (apartado 5.6). De este modo, se puede observar que el menor porcentaje de tiempo empleado corresponde con el tiempo que se necesita para enviar los mensajes a través de la red, e incluso en algunos casos es despreciable, ya que su valor es 0 debido a la precisión utilizada para medirlo, lo cual significa que el tiempo necesitado será menor a 1ms. Era de prever, puesto que todos los componentes del entorno de pruebas están conectados directamente.

Se puede estimar, que evidentemente el tiempo de incomunicación total de un MN en un entorno real, en el que los nodos no estén conectados directamente sea mayor. La diferencia, residirá en el retardo de envío de mensajes.

Por tanto, se concluye que los objetivos fijados al inicio del proyecto han sido cumplidos, dado que finalmente se ha conseguido una implementación de PMIPv6. Cabe destacar que dicha implementación es altamente estable y robusta, puesto que siempre se consigue restablecer la comunicación de todos los elementos del entorno de pruebas en un tiempo aceptable. De esta forma, las expectativas iniciales de la implementación han sido satisfechas. Esto ha sido demostrado a lo largo del presente documento, tanto cualitativa como cuantitativamente.

6. 2 Líneas de trabajo futuro

A partir del trabajo realizado a lo largo del proyecto se podrán realizar más trabajos sobre el mismo tema, e incluso, tomándolo como base. Existen ciertas funcionalidades no desarrolladas en la implementación realizada. Entre otras, cabe destacar:

- Incluir comprobaciones relativas al número de secuencia o al sello temporal incluido en los mensajes PBU o PBA.
- Incluir en los PBA estados de aceptación, es decir, facilitar información al MAG de cómo se ha registrado un nodo, o de porqué ha sido rechazado.
- Añadir comprobaciones relativas al tiempo de vida de una determinada entrada y cuando antes de que ese tiempo haya vencido, actualizar dicha entrada, enviando los mensajes necesarios.

Con respecto a otras funcionalidades, no incluidas específicamente en [5], pero que sí serían deseables que la implementación contase con ellas, se pueden mencionar como más importantes:

- Dotar de seguridad a todos los mensajes enviados con motivo de la señalización perteneciente e PMIPv6. En [5], es recomendado realizarlo mediante IPsec [6].
- Extender las variables de entrada en el fichero de configuración, para poder dar la posibilidad de una configuración más personalizada. Esto puede ser realizado, como se realizó en la primera implementación con la que se trabajo, mediante la definición de una gramática asociada a dicho fichero.
- Además, sería altamente aconsejable, introducir en la implementación una optimización de rutas. Esto podría ser realizado mediante la creación de varias tablas de rutas en cada nodo, y dependiendo del origen y destino del paquete, utilizar una tabla u otra.

En cuanto al plan de pruebas propuesto, se podría realizar uno más exhaustivo. Debería ser probado también en un entorno más real, y por lo tanto más complejo, incluyendo más nodos intermedios. Además, deberá ser validado cuando más de un MN esté usando los servicios de PMIPv6 facilitados por esta implementación.

Por último serán necesarias pruebas de interoperabilidad del sistema desarrollado con otros sistemas que lleven a cabo las mismas funciones para corroborar que la implementación cumple con el protocolo PMIPv6, y que es capaz de realizar las mismas acciones tras percatarse de los mismos eventos, e intercambiar los datos pertinentes con las demás entidades del entorno.

CAPÍTULO

7

ANEXOS

Que algo no funcione como tú esperas no quiere decir que sea inútil.

Thomas Alva Edison (1847-1931)

A. Funcionamiento detallado de PMIPv6

I. Seguridad en PMIPv6

PMIPv6 deberá asegurar que el intercambio de mensajes producido entre el LMA y el MAG mantenga su integridad durante el trayecto, así como una autenticación del origen de los datos. Para conseguirlo, tanto el MAG como el LMA deberán implementar IPsec [6]. Con este protocolo se consigue proteger el tráfico enviado por el túnel bidireccional entre el LMA y el MAG. Con ello, el protocolo será seguro ante ataques de suplantación de personalidad del MAG o del LMA.

El uso del protocolo IKEv2 (Internet Key Exchange protocol version 2), podría servir para proteger los PBUs y PBAs, además de que tanto el LMA como el MAG usen mecanismos de autenticación mutua.

El nodo móvil no tiene porqué aportar ningún mecanismo de seguridad, por lo tanto, será el LMA el que restrinja la creación y manipulación de las asociaciones intermedias. Para poder llevar a cabo esta tarea el LMA previamente ha de ser configurado para ello.

Otros posibles ataques que se podrían perpetrar en este tipo de redes serían por ejemplo, la denegación de servicio o el "secuestro" de una sesión de movilidad, por ello se exige que el LMA sólo permita a MAGs autorizados por el dominio PMIPv6 en el que se encuentran enviar PBUs en nombre del nodo móvil.

Para evitar posibles ataques en el intercambio de mensajes entre el MAG y el MN, es requerida una autenticación del nodo siempre que se una a un nuevo link, y que el MAG lo autorice a formar parte de él. Esto asegurará que las cachés contendrán la información adecuada. Además, el MAG autentica todos y cada uno de los mensajes que recibe comprobando el identificador del nodo.

II. Operaciones del LMA

Esta entidad no sólo ha de realizar las mismas tareas realizadas por el *home agent* de Mobile IPv6, sino que también debe realizar nuevas tareas propias de PMIPv6.

El LMA tiene que mantener una *binding cache* con la información de todos los nodos a los cuales gestiona su movilidad. Por cada nodo tendrá una entrada diferente con su

información actualizada. Algunos campos de cada entrada de esta *binding cache* son los mismos utilizados en Mobile IPv6, estos son [4]:

- *Home Address* del MN.
- *Care-of-address* del MN. En PMIPv6 corresponde con la *Proxy care-of-address.*
- Tiempo de vida de la entrada del nodo. Determina el tiempo restante de validez de la entrada en la *binding cache.*
- Número de secuencia más alto recibido hasta el momento con relación a ese nodo.
- Información relacionada con la política de uso de esa entrada. Uno de los usos típicos de este campo es el de saber cómo se ha de realizar el reemplazo de la entrada cuando se reciba una nueva actualización.

Los campos añadidos a cada entrada para poder llevar a cabo las nuevas funcionalidades de PMIPv6 corresponden con:

- Un *flag* (P-flag) que determina si la correspondiente entrada ha sido creada debido a un *proxy* (normalmente MAG). En tal caso su valor será igual a uno, en caso contrario, es decir lo haya dado de alta otra entidad su valor será 0.
- Identificador del nodo móvil.
- Identificador de la capa de enlace del nodo móvil.
- Dirección del link que el MAG comparte con el nodo móvil.
- Lista de prefijos asignados a la interfaz por la que está conectado actualmente el nodo móvil.
- Identificador del túnel bidireccional entre el LMA y el actual MAG del nodo móvil.
- Tipo de tecnología de acceso, por la cual el nodo está conectado a la interfaz.
- Valor más reciente del sello temporal, es decir la hora local del LMA cuando se recibió el último mensaje relativo a ese nodo. Más adelante, en esta misma sección, este campo es especificado con más detalle.

Puede haber más de un prefijo asignado a una única interfaz del nodo móvil, pero cada uno de esos prefijos debe pertenecer únicamente a ese nodo móvil, y todos ellos son parte de exactamente una sesión de movilidad. Si el nodo se conecta al dominio PMIPv6 a través de varias interfaces simultáneamente, a cada una de ellas se le asignará al menos un prefijo único. Dos prefijos asignados a diferentes interfaces nunca podrán ser manejados dentro de una misma sesión de movilidad. Estos prefijos residirán en el link de acceso del nodo móvil al dominio PMIPv6, aunque desde el punto de vista de encaminar los mensajes desde y hacia el nodo móvil, residirán en el LMA.

Habitualmente, cualquiera de los prefijos asignados a la interfaz por la que el nodo está conectado al dominio PMIPv6 es un criterio de búsqueda para localizar la entrada del nodo móvil en la *binding* caché. En el caso en el que se ha producido un *handoff* a un nuevo MAG, y éste nuevo MAG desconoce los prefijos asignados a tal sesión de movilidad, no será viable realizar la búsqueda mediante este criterio.

A continuación se muestran una serie de consideraciones que han de ser respetadas en el momento de procesar los PBUs:

- Cuando el P-flag tiene valor uno, entonces se deberá llevar a cabo la autenticación del nodo origen del PBU.
- A la hora de procesar las cabeceras de movilidad, el LMA deberá seguir las siguientes normas [4]:

    - El checksum ha de ser verificado. En caso de no validarse se ha de rechazar el mensaje.
    - El tipo de la cabecera de movilidad (*MH type*), deberá corresponder con el valor 0. En caso de ser un valor con significado desconocido, el LMA descartará el mensaje automáticamente.
    - El campo "payload proto" contendrá *"IPROTO_NONE"*, que corresponde con el valor 59 en decimal. En caso contrario se rechazará el mensaje y se enviará un mensaje ICMP al origen del PBU.
    - La longitud de la cabecera de movilidad será siempre múltiplo de 8. Su longitud dependerá de las opciones que lleve el paquete.

- Del mismo modo, el LMA deberá ignorar las comprobaciones relativas a la *care-of-address* especificadas en [4].
- El nodo móvil ha de ser identificado por el LMA mediante su identificador, en caso de que el mensaje no contenga dicho campo, será rechazado, y enviará al origen del PBU un PBA indicando el motivo del rechazo.
- Puede darse la circunstancia que el PBU sí contenga identificador del nodo, pero que el LMA no sea capaz de identificarlo. En cuyo caso se rechazará dicho PBU y se enviará el correspondiente PBA comunicando el problema.
- Una vez recibido un PBU se debe aplicar la política pertinente para realizar las comprobaciones oportunas.
- Si se determina que la entidad origen del PBU no es una entidad autorizada, se ha de descartar dicho mensaje, enviando la correspondiente notificación mediante un PBA.
- En caso de que el LMA no autorice la movilidad del nodo, el PBU debe ser descartado, y enviará un PBA indicándolo.
- En el caso que el PBU no contenga el campo de opciones de los prefijos de la *Home Network,* el mensaje será rechazado enviando un PBA anunciando el porqué.
- Si el indicador del *Handoff* no se encuentra presente, el LMA descartará el PBU, y enviará un PBA informando de ello.
- En caso de no encontrarse el tipo de tecnología utilizada por el nodo para acceder al dominio PMIPv6, el LMA rechazará el PBU mandando un PBA.
- Se ha de comprobar la existencia de una entrada en la *binding cache.* correspondiente al PBU recibido, en caso de no existir dicha entrada se deberá crearla.

Con respecto a la creación de una nueva sesión de movilidad, es decir de registrar por primera vez una asociación de un nodo, se ha de tener en cuenta que:

- Si al menos una instancia del campo de opciones de los prefijos de la *home network* no tiene ningún valor asignado, es decir tiene 0 en todos los bytes del campo, entonces el LMA deberá asignar al menos un prefijo a dicha sesión de movilidad y asegurar que ese prefijo no está siendo usado por ninguna otra sesión.
- En el caso de que no sea capaz de asignar ningún prefijo, entonces el LMA deberá rechazar la solicitud enviando el correspondiente PBA.
- Si existiesen instancias de prefijos con valores distintos a 0, el LMA, antes de aceptar la petición, deberá comprobar uno a uno que dichos prefijos son únicos y que además el nodo está autorizado a usarlos. En caso de no cumplirse una de estas condiciones se descartará la solicitud y se enviará un PBA anunciando el motivo.
- Una vez admitida la solicitud se debe crear la entrada en la *binding cache* con los valores aceptados.
- Si no existiese un túnel bidireccional entre el LMA y el MAG que envió el PBU, será el LMA el encargado de crearlo, siempre y cuando se haya aceptado la solicitud.
- El LMA debe crear también, el prefijo para encaminar por el túnel bidireccional todo el tráfico recibido de los prefijos de la *home network* del nodo móvil hacia el MAG correspondiente.
- Una vez terminadas todas las tareas anteriores, el LMA enviará al MAG un PBA en el cual el campo estado contendrá un 0, lo cual indica que se ha aceptado la petición.

En el caso de recibir un mensaje PBU para alargar el tiempo en el que la *binding cache* mantiene activa una entrada, el LMA deberá comprobar que lo ha recibido del mismo MAG que envió la última actualización para ese nodo móvil, es decir el campo *Proxy-CoA* en la petición es la misma que en la *binding cache*. En tal caso, se actualizará al valor indicado en el PBU. Una vez actualizado, se enviará un PBA con el campo estado igual a 0 para indicar la aceptación del PBU.

Puede suceder que se reciba un PBU para ampliar el tiempo que va a permanecer una determinada entrada en la *binding cache* en el que no coincida el campo *Proxy-CoA* de la petición con el registrado, en cuyo caso puede haberse producido un *handoff.* De ser así, la LMA deberá realizar las siguientes tareas:

- Eliminar la ruta asociada a la anterior sesión de movilidad con los prefijos de la *home network* del nodo móvil.
- Destruir el túnel bidireccional con el MAG asociado a la mencionada sesión de movilidad en caso de que ningún otro nodo tenga asociada esa MAG.
- Si no existe túnel bidireccional que una el LMA con el MAG que envía la petición, ha de ser construido.

- A continuación se debe crear la ruta para encaminar los mensajes con los prefijos de la *home network* del nodo móvil por el nuevo túnel que une el LMA con el nuevo MAG asociado al nodo móvil.
- Por último se envía al nuevo MAG un PBA anunciando la correcta recepción del PBU estableciendo el campo estado a 0.

Si un MAG quiere anunciar al correspondiente LMA que uno de los nodos móviles que él gestionaba ya no se encuentra en su link de acceso deberá enviar un PBU a dicho LMA que contenga el valor 0 en el campo tiempo de vida. En el caso de recibirse un PBU de este tipo, se deberá comprobar que la dirección origen del paquete es igual a la dirección que se encuentra en el campo *Proxy-CoA*; de no ser así el paquete será rechazado. Si no es rechazado entonces, el LMA deberá esperar un período de tiempo antes de que dicha entrada sea borrada de la *binding cache*. Durante ese período:

- El LMA omitirá el tráfico correspondiente al nodo móvil.
- Si el LMA recibe un PBU con el campo tiempo de vida mayor que 0, y la petición es aceptada, no borrará la entrada de la sesión de movilidad si no que actualizará sus campos con los valores obtenidos del PBU.
- En caso de no recibir ningún PBU asociado a ese nodo, una vez que finalice el tiempo de espera, la entrada deberá ser borrada de la *binding cache*, pudiendo ser los prefijos reasignados a otros nodos por el LMA.

Se ha mencionado con anterioridad algunos detalles de cómo debe ser construido un PBA en el LMA, como por ejemplo el valor que se le asigna al campo estado. A continuación se muestran todos los campos que pueden ser portados por la cabecera de movilidad IPv6 y sus posibles valores cuando se envía un PBA:

- **Dirección origen:** corresponderá con la dirección destino del PBU.
- **Dirección destino:** será la dirección origen enviada en el PBU. También coincidirá con la dirección del campo *Proxy-CoA.*
- **Identificador del nodo móvil:** debe estar siempre presente. Este campo será copiado del campo homónimo del PBU.
- **Prefijo de Home Network:** al menos debe existir un prefijo de la red. Se copiarán de los prefijos enviados en el PBU. En caso de que la petición sea rechazada no será necesario enviar todos los prefijos, bastará con indicar que había prefijos, dándole el valor 0 a este campo.
- **Indicador de Handoff:** este indicador debe ser copiado del PBU. En caso de no encontrarse deberá ser establecido con valor 0.
- **Tipo de la tecnología de acceso:** sucede lo mismo que con el campo anterior.
- **Sello temporal:** esta opción deberá estar presente siempre y cuando lo estuviese en el PBU recibido.
- **Identificador del link del nodo móvil:** únicamente se encontrará en el PBA si estaba presente en el PBU. En tal caso, se copiará del PBU.

- **Dirección del link:** al igual que en los casos anteriores estará presente si y sólo si lo estaba en el PBU. Si se rechaza la petición, en el PBA irá el mismo valor que se encontraba en el PBU. Si es aceptado, entonces se tiene en cuenta que:
  - Si el valor del campo en el PBU es igual a 0, y además existe una entrada en la *binding cache* asociada a este PBU, entonces se copiará el valor que contenga dicha entrada para este campo.
  - Si el valor del campo en el PBU es igual a 0, pero no existe ninguna entrada asociada a tal PBU, el LMA deberá generar una dirección válida para ese link para que el MAG se pueda comunicar con el nodo móvil. La dirección generada será la enviada en este campo.
  - Si el valor del campo en el PBU es distinto a 0, entonces la dirección de él, será copiada en el PBA de respuesta.

Los principales aspectos a tener en cuenta, y que debe llevar a cabo el LMA, para que un nodo móvil pueda conectarse a través de múltiples interfaces a un mismo dominio PMIPv6 son:

- Adjudicar a cada interfaz una sesión de movilidad diferente.
- Cada interfaz puede tener más de un prefijo asociado, pero todos los prefijos de una misma interfaz serán gestionados en una única sesión de movilidad.
- Cuando se produce un *handoff* del nodo, los prefijos de la anterior interfaz utilizada por el nodo, pueden ser asignados a la nueva interfaz.

Por lo tanto, puede haber varias entradas relacionadas con un mismo nodo dentro de una misma *binding cache.* A la hora de procesar un PBU recibido en el LMA, suele ser necesario realizar búsquedas en la *binding cache*, y para encontrar la entrada deseada habrá que seguir las siguientes normas dependiendo de los siguientes supuestos:

- Al menos un prefijo de *home network* presente en la petición con valor distinto de 0:
  - Si se encuentra más de un prefijo en la petición, todos y cada uno de ellos pueden ser usados para localizar la entrada.
  - LMA comprueba que en la *binding cache* exista una entrada que utilice uno de los prefijos indicados en el PBU.
  - En caso de que tal entrada no exista, entonces se creará una nueva entrada para almacenar el nodo asociado con la petición.
  - Si existe una entrada en la que se usen los mismos prefijos que los que se reflejan en la petición, el identificador del nodo ha de ser comprobado. En caso de no coincidir el de la *binding cache* con el del PBU el LMA rechazará la petición enviando el correspondiente mensaje PBA.
  - En caso de coincidir el identificador del nodo, si no coinciden todos los prefijos, tanto en número como en valor, del PBU con los almacenados en la entrada de la *binding cache*, entonces el LMA descartará la petición y emitirá el correspondiente PBA.

- Si coinciden todos los prefijos de la entrada de la *binding cache*, y el identificador del nodo, entonces se actualizará la *binding cache,* siempre y cuando al menos una de las siguientes condiciones es verdadera:

    - Han de coincidir los valores de la petición y de la entrada de la *binding cache* tanto el identificador del link como el tipo de la tecnología de acceso.
    - El valor del indicador de *handoff* en el PBU está establecido a 2, es decir, se ha producido un *handoff* entre dos interfaces del nodo móvil.
    - No está presente el identificador del link en la petición, en la *binding cache* su valor es 0, el tipo de la tecnología de acceso de la petición coincide con el de la *binding cache*, y además el indicador del *handoff* es igual a 3 (se ha producido *handoff* entre MAGs pero con la misma interfaz del nodo móvil).
    - Deben coincidir los valores, tanto en la petición como en la *binding cache*, de los campos de *Proxy-CoA* y tipo de tecnología de acceso.

- Para el resto de los posibles casos, el mensaje será considerado como de creación de una nueva entrada en la *binding cache.* Si el valor del campo tiempo de vida es igual a 0, o no es posible crear la entrada correspondiente con la petición, entonces ésta será ignorada.

- No se encuentra en la petición ningún prefijo de la *home network*, o los que hay tienen valor 0, pero sí se dispone del identificador del link del nodo móvil:

    - El LMA debe comprobar si los valores de una entrada de los campos de identificador de nodo, tipo de tecnología de acceso e identificador del link coinciden con sus respectivos campos en el PBU.
    - Si existe, entonces el mensaje de petición sirve para que dicha entrada se actualice en la *binding cache.*
    - Si no se halla dicha entrada y el campo indicador de *handoff* es igual a 2 (se ha producido handoff entre dos interfaces diferentes del nodo), entonces, el LMA debe comprobar que existe una, y sólo una entrada, en la cual el identificador del nodo tenga el mismo valor que el campo equivalente en la petición. En tal caso, la petición será de actualización.
    - En caso de no existir ninguna entrada en la que los valores del identificador del nodo, el tipo de tecnología de acceso y el identificador del link coincidan simultáneamente en la *binding cache* y en la petición, y el indicador de *handoff* tiene valor 4 (estado desconocido), el LMA deberá comprobar que existe una única entrada

con valor igual al del PBU en el campo de identificación del nodo, indistintamente del valor del link. En caso afirmativo, debería esperar hasta que se produzca la baja de dicha entrada para considerar la petición como un PBU de actualización. Si no se recibiese un mensaje para dar de baja dicha entrada en un período de tiempo determinado, entonces considerará el mensaje de petición PBU como un mensaje para la creación de una nueva sesión de movilidad del nodo. Esta nueva sesión podría ser creada sin esperar dicho período de tiempo, según la elección del LMA.

- Al igual que en el supuesto anterior, los demás casos deberán ser considerados como peticiones de creación de una nueva sesión de movilidad, a no ser que el campo tiempo de vida tenga un valor 0, o no sea posible su creación. En tal caso la petición será ignorada.

- No se encuentran ni el identificador del link, ni ningún prefijo con valor distinto a 0 presentes en la petición:

  - El LMA deberá comprobar la unicidad de la entrada de la *binding cache* cuyo identificador del nodo sea igual al identificador del nodo presente en la petición.
  - Si sólo existe una entrada, y el valor del indicador del *handoff* 2 ó 3, entonces directamente la petición se considerará como una actualización de dicha entrada.
  - Si sólo existe una entrada, pero el valor del indicador del *handoff* es 4, el LMA deberá esperar hasta que se reciba una petición de eliminación de dicha entrada antes de proceder a la actualización. Si tal petición no se recibe se debe considerar como que el PBU se recibió inicialmente con motivo de crear una nueva sesión de movilidad. Puede que el LMA decida no esperar dicho tiempo y crearla inmediatamente.
  - Al igual que en los supuestos anteriores, en los demás casos posibles dentro de este supuesto, el PBU se considerará siempre que se ha recibido con el fin de crear una nueva sesión de movilidad, a excepción de que el campo tiempo de vida de la petición sea igual a 0 o que no se localice la entrada correspondiente en la *binding cache*, en cuyo caso se ignorará.

No resulta viable utilizar el esquema para determinar el número de secuencia usado en MIPv6 en este nuevo protocolo, dado que el nodo móvil va desplazándose de MAG en MAG, y no existen mecanismos para transmitir el contexto de un MAG al otro. Si el LMA no es capaz de ordenar los mensajes según su antigüedad cabe una posibilidad muy alta de que la *binding cache* no se actualice correctamente. Para resolver este problema se pueden adoptar dos soluciones:

I. Sellos temporales:

El principio básico para el funcionamiento de este esquema, consiste en que el nodo que envíe el mensaje inserte la hora de envío en el mensaje, y el que recibe dicho mensaje, para conocer qué mensaje es más reciente únicamente deberá comparar este campo.

Este esquema, al igual que otros tiene una serie de inconvenientes:

- No podrá usarse cuando alguno de los nodos no posea reloj para poder insertar la hora en el mensaje.
- El tiempo de reconexión es el tiempo total que un nodo móvil viaja de un MAG a otro más el tiempo en el que este segundo MAG se percata de que un nuevo nodo se ha unido a su link. Si la diferencia de hora entre ambos relojes (desincronización) es más de la mitad del tiempo de reconexión, este esquema no funcionará adecuadamente.

Para un correcto funcionamiento de esta solución se han de seguir las siguientes directrices:

- El LMA, obligatoriamente ha de soportar está opción, y si en el PBU se encuentra un sello temporal, en su respuesta, PBA, también ha de añadirse un sello válido generado por él.
- Los relojes de las entidades que intercambian mensajes deberán estar sincronizados para un correcto funcionamiento.
- Además, los relojes de dichas entidades deberían estar sincronizados con un recurso común. También es posible que sea el nodo móvil el que genere el valor del sello temporal, y que el MAG lo utilice, siempre y cuando dicho valor se vaya incrementando, para así no depender de una fuente externa.
- Cuando se genera un valor para incluir el sello temporal en el mensaje, la entidad que lo genere debe asegurarse que dicho valor es correcto.
- Si el sello temporal está presente en el PBU, el LMA deberá ignorar el número de secuencia, en caso de que se encuentre presente también en el mensaje.
- Una vez que el LMA reciba un mensaje con la opción de sello temporal, deberá validar su valor, para ello deberá cumplir que:
    - El valor del sello temporal deberá ser un valor cercano a la hora del LMA.
    - El valor del sello temporal del mensaje que se acaba de recibir ha de ser mayor al recibido anteriormente.

- Si el valor del sello temporal es válido, entonces el LMA deberá enviar al nodo móvil el mismo valor de sello temporal que le envíe al MAG.

- Si el valor del sello temporal del último PBU es menor que el valor del mismo campo de otro PBU recibido con anterioridad relativo a la misma sesión de movilidad, entonces el último recibido ha de ser rechazado. En el PBA que envíe anunciándolo ha de incluir un sello temporal con el valor actual.
- En general, si el valor del sello temporal no es válido, el PBU será rechazado, indicando en el PBA el por qué y añadiendo el valor actual de la hora.

II. Número de secuencia:

Para que esta solución funcione adecuadamente, el MAG del nodo móvil ha de ser capaz de conseguir el último número de secuencia que ha sido enviado para esa sesión de movilidad. Para ello, el número de secuencia ha de ser mantenido por el nodo móvil y ha de poder comunicárselo al MAG al que se asocie.

Para que este esquema funcione adecuadamente:

- El valor del número de secuencia solamente se tendrá en cuenta cuando la opción de sello temporal no esté presente en el mensaje. En el mensaje de contestación, no será necesario, por tanto, incluir el sello temporal.
- Se debe aceptar el campo número de secuencia tal y como se especifica en [4].
- Esta solución únicamente podrá ser usada si existe un mecanismo para que el MAG pueda recuperar el último nodo de secuencia que se envió con relación a la sesión de movilidad del nodo móvil.

Para poder encaminar el tráfico de mensajes relativo al nodo móvil se utiliza el nombrado túnel bidireccional, que conecta el LMA y el MAG, para que así el nodo pueda utilizar la/s dirección/es de su *home network*. Este túnel esconde la topología de la red que se encuentra entre ambas entidades, por lo que no se conocerá los nodos que pueda haber entre una entidad y otra. A la hora de establecer el túnel, se pueden utilizar túneles previamente establecidos, o lo que es lo mismo, estáticos, pero es altamente aconsejable utilizar túneles dinámicos, para así poder crearlos cuando se necesiten y destruirlos cuando ya no sean necesarios. En tal caso, se deberá seguir las siguientes indicaciones:

- El túnel se establece entre el LMA y el MAG, por lo que los extremos de dicho túnel serán la dirección del LMA y la *Proxy-CoA*, respectivamente.
- Varias sesiones de movilidad pueden usar el mismo túnel. Al igual que las sesiones, se debe establecer un tiempo de vida de dicho túnel, que será

actualizado con cada actualización de las sesiones correspondientes, y además mantener un contador para saber cuántos nodos lo están usando.
- Se eliminará el túnel cuando expire el tiempo de vida, o cuando no esté siendo utilizado por ningún nodo.

Un LMA que gestione la movilidad de un nodo móvil recibirá todos los paquetes cuya dirección destino sea una de las direcciones de la *home network* de dicho nodo. Cuando se reciba un paquete para el nodo móvil, entonces deberá facilitar una ruta para el encaminamiento de dicho mensaje. Deberá realizar algo similar si por el contrario recibe un mensaje cuyo origen sea el nodo móvil. Para ello, se distinguen ambos casos:

- Paquetes recibidos con destino el nodo móvil:
  - El LMA enviará el mensaje por el túnel establecido con el MAG creado para dicho nodo.
  - El paquete enviado por el túnel deberá estar encapsulado en IPv6, y deberá ser de un formato específico definido en la sección 5.6.2 de [5].
- Paquetes recibidos con el nodo móvil como origen:
  - Los mensajes recibidos por el túnel desde el MAG, han de ser enviados a la dirección destino una vez se hayan eliminado las cabeceras pertenecientes al enrutamiento a través del túnel.

III. Operaciones del MAG

Esta entidad será la responsable de detectar cuando un nuevo nodo se ha unido a su link, y de enviar un PBU al LMA correspondiente, es decir, gestiona la movilidad del nodo en su lugar. Esta funcionalidad se suele desarrollar en el *router* de acceso, aunque puede ser compartida por múltiples sistemas. Por lo tanto, tendrá que llevar a cabo:

- Detectar movimiento, tanto de unión como de desunión en su link de acceso, y comunicárselo al correspondiente LMA.
- Simular que el nodo móvil se encuentra en su *home network* continuamente.
- Crear lo que sea necesario para facilitar al nodo móvil el establecimiento de sus prefijos de su *home network*.

Para ello, deberá mantener una B*inging Update List* (BUL), igual que en Mobile IPv6, en la cual se almacenará una entrada por cada sesión de movilidad de un determinado nodo asociado al LMA correspondiente. Para ello, la BUL de Mobile IPv6 deberá ser extendida con los siguientes campos:

- Identificador del nodo móvil.
- Identificador del link de la interfaz por la cual el nodo está conectado. Suele ser facilitado por el *router* de acceso cuando el nodo solicita un *router* por defecto. Si no existe, su longitud será de 2 octetos, y estará inicializado con valor.
- Lista de prefijos de la *home network* del nodo móvil.
- Dirección del link de acceso al MAG.
- Dirección IP del LMA que gestiona la movilidad del nodo.
- Identificador de la interfaz por la que el nodo se ha conectado al link del MAG.
- Identificador del túnel bidireccional que une dicho MAG con el LMA.

El perfil o política del nodo móvil define los parámetros operacionales para que las entidades puedan gestionar correctamente la movilidad del nodo. Dichos parámetros deben ser obtenidos por el MAG y el LMA. Para que el MAG pueda desarrollar su funcionalidad correctamente, necesitará los siguientes parámetros, de los cuales los dos primeros serán obligatorios y los demás opcionales, pero altamente recomendables:

- Identificador del nodo.
- Dirección IP del LMA.
- Prefijos de la *home network* del nodo móvil.
- Tiempo de vida de dichos prefijos.
- Procedimientos para la configuración del nodo móvil.

PMIPv6 únicamente soporta el tipo punto-a-punto (point-to-point) en los links de acceso, es decir, que el MAG y el nodo móvil sean los únicos nodos en dicho link. Se podrían utilizar otros tipos de links de acceso siempre y cuando se emule al mencionado.

En el momento en el que un nodo móvil se une al link del MAG, los mecanismos de seguridad deberán autenticar y autorizar al nodo móvil, y sólo entonces será cuando el MAG le ofrezca sus servicios.

Todo nodo móvil debe poder ser identificado por toda entidad dentro de un dominio PMIPv6, de ahí la necesidad de un identificador. Éste debe invariable y único en dicho dominio. Algunas observaciones sobre este campo que han de ir presente tanto en PBU como en PBA son:

- Típicamente, el identificador del nodo es obtenido en el proceso de autenticación. El identificador de usuario no valdría como identificador del nodo, dado que un mismo usuario se podría conectar desde diferentes nodos, y por tanto no serían identificados inequívocamente.
- En algunos casos el identificador obtenido puede ser temporal, por lo que podría cambiar. Incluso si esto sucede, el MAG debe poder usar este identificador para obtener todos los datos necesarios sobre el nodo.
- Por motivos de seguridad, es posible que un nodo utilice un identificador falso, e igual que antes, el MAG debe ser capaz de usarlo.

- Una vez que el MAG haya identificado el nodo móvil, entonces deberá asociarlo con el link por el cual ha tenido acceso a él.

Una de las responsabilidades del MAG que se ha comentado anteriormente, es que éste ha de ser capaz de emular a la *home network* del nodo móvil, y así que éste último no detecte que ha cambiado su ubicación y con ello el link de acceso al dominio. El MAG puede conocer los prefijos de la *home network* o bien enviando un *router advertisement* o bien a través de los PBA recibidos o bien por el perfil de la política del nodo móvil. Una vez conocidos, el MAG les asignará un tiempo de vida a cada uno de ellos.

Aunque un nodo se desplace entre un MAG y otro, él seguirá detectando su *home network* y por lo tanto el mismo link de acceso al dominio. Pese a ello, cada vez que él cambie de link de acceso, se lanzará un mecanismo para comprobar si su/s dirección/es están duplicadas. Realizar dicha comprobación es relevante, dado que el nodo móvil y el MAG pueden estar configurados con prefijos pertenecientes a la misma red, y por lo tanto producirse colisiones entre direcciones. Para evitar dicho problema, se sugiere que la dirección de link de acceso que configura el MAG punto-a-punto con el nodo móvil sea generada por el LMA. Es deseable además que antes de que el nodo complete su fase de configuración, la dirección del link sea facilitada previamente al MAG.

Al igual que sucedía con el LMA, cuando se realiza el intercambio de mensajes, en el MAG se han de tener unas ciertas consideraciones, dependiendo de los diferentes casos que se puedan producir, para el correcto funcionamiento del protocolo. Por ello, en los casos:

- **Registro inicial de un nodo:**

  - Inmediatamente después de que el MAG detecte que un nuevo nodo se ha unido a su link, debe identificarlo y determinar si se le ofrecerá servicio de movilidad, en cuyo caso le enviará un PBU al LMA correspondiente.
  - El PBU siempre deberá incluir el identificador del nodo.
  - También deberán incluirse los prefijos de la *home network* del nodo móvil, en caso de conocerse. Si se incluye un único prefijo con valor 0, entonces el MAG preguntará al LMA por dichos prefijos.
  - El indicador de *handoff* debe estar presente en el PBU y tomará distintos valores dependiendo de la situación:

    - Tomará valor 1 siempre y cuando el MAG determine que no se ha producido un *handoff* de una sesión de movilidad existente por el cual se haya unido a su link. Indicará principalmente que el LMA deberá crear una nueva entrada en la *binding cache*.
    - Será establecido a 2, si el MAG sabe que el nodo móvil se ha unido a su link debido a un *handoff* entre dos interfaces diferentes del nodo.

- Si su valor es de 3 será porque el MAG haya concluido que el nodo móvil se ha unido a él al realizar un *handoff* entre MAGs, pero manteniendo la misma interfaz.
- Tomará el valor 4 si el MAG no puede determinar el porqué de la unión de ese nodo a su link.

- O bien el sello temporal o bien un número de secuencia válido debe estar presente obligatoriamente en el PBU. Es recomendable que aunque se añada el sello temporal siempre se incluya el número de secuencia.
- El identificador de la interfaz por la que actualmente está unido al link de acceso también ha de estar presente en el mensaje siempre que se conozca. En caso contrario este campo tomará el valor 0.
- El tipo de la tecnología de acceso al MAG es obligatorio añadirlo en el PBU.
- La dirección del link por el que el nodo se ha conectado al MAG deberá estar presente siempre que una determinada variable tenga valor 0. Cuando es 0, será porque el MAG esté preguntando al LMA por dicha dirección.
- En caso de no existir una entrada en la BUL de ese MAG para el nodo móvil, el MAG por lo tanto, deberá crearlo una vez se haya enviado el PBU.

- **Recepción de un PBA:**

  - Si el *flag* P está establecido a 1, entonces debe ser autenticado mediante IPsec.
  - A la hora de procesar las cabeceras del PBA se deberán seguir las reglas de [4] para el tratamiento de las cabeceras de movilidad y explicadas en la sección anterior.
  - El número de secuencia o el sello temporal ha de ser tratado como se trataban en la sección anterior.
  - El MAG deberá ignorar cualquier comprobación a realizar sobre la cabecera de enrutamiento tipo 2, definida en [4] como una variante de la cabecera de enrutamiento para permitir al paquete ser encaminado desde cualquier nodo al *care-of-address*.
  - El identificador de nodo presente en el PBA ha de ser comprobado, para asegurarse de que corresponde con un nodo del que se haya enviado PBU. Si existen varios mensajes con el mismo identificador será porque se han enviado varios PBU, y el número de secuencia o el sello temporal ayudará a decidir qué respuesta es asociada con qué petición. En caso de no haberse realizado ninguna petición con ese identificador de nodo el mensaje será rechazado.
  - Si el valor de los campos indicador de *handoff*, tipo de tecnología de acceso, identificador de link, o identificador del nodo del PBA no coincide con el del correspondiente PBU, la petición se ignorará. No se volverá a retransmitir la petición realizada anteriormente.

- Si el campo estado del PBA corresponde con el valor "PROXY_REG_NOT_ENABLED", el registro del nodo móvil no será posible en el LMA y no se debería volver a mandar el PBU. Además la gestión de movilidad para ese nodo debe ser denegada.
- En cambio, si el campo estado tiene valor "TIMESTAMP_LOWER_THAN _PREV_ACCEPTED", es decir, que el sello temporal recibido en el PBU es menos que otro PBU para el mismo nodo que se aceptó con anterioridad, se debería intentar registrar cuando reafirme la presencia del nodo en el link. No es necesario sincronizar el reloj del MAG tras este error.
- En el caso que el campo estado contenga el valor "TIMESTAMP_MISMATCH", el MAG debería volver a intentar registrar ese nodo en el LMA después de haberse sincronizado con el reloj común a todas las entidades del dominio PMIv6.
- Si el PBA recibido tiene el campo estado establecido a "NOT_AUTHORIZED _FOR _HOME_NETWORK_PREFIX", significará que no le estará permitido al nodo móvil utilizar uno o varios de los prefijos de *home network* que se ha enviado en el PBU, por lo que el MAG no debería proceder a registrar ese nodo con los mismos prefijos otra vez. Puede solicitar que sea el LMA quién le asigne los prefijos a ese nodo.
- En caso de que el valor estado en cualquier PBA recibido sea mayor o igual a 128, el LMA habrá rechazado la petición, y el MAG no se lo comunicará al nodo, simplemente le denegará la funcionalidad de movilidad, y puede incluso que el MAG abandone el link en el cual el nodo estaba unido a él.
- Por el contrario, si el campo estado tiene un valor de 0, el MAG establecerá el túnel bidireccional con el LMA, ya que la petición habrá sido aceptada.
- Una vez aceptada la petición, el MAG también deberá establecer la ruta por la cual los mensajes procedentes de cualquier dirección dentro de los prefijos de un nodo móvil, sean enviados hacia el LMA a través del túnel.
- Posteriormente, actualizará su BUL con la nueva información relativa al nodo.
- En caso de que el PBA recibido tenga el campo dirección del link establecido a 0, será el MAG el encargado de configurar el link punto-a-punto. No deberá configurarlo sin usar DAD, lo que evitará en gran medida la coincidencia de direcciones en dicho link. Si la dirección que genera el MAG está siendo usada en ese mismo link no deberá usar esa dirección y configurar otra diferente. Una vez configurada deberá comunicárselo al LMA.

- **Ampliación del tiempo de vida de las entradas:**

  - Para extender el tiempo de vida de una entrada en la *binding cache* del LMA, bastará simplemente con enviar una actualización, es decir un PBU, claro está que deberá ser después de haber registrado el nodo con éxito.

- El PBU de actualización deberá contener una opción de prefijo *home network* por cada uno de los prefijos asignados a dicha sesión de movilidad. Evidentemente, cada campo contendrá el valor del correspondiente prefijo.
- El valor del indicador de *handoff* será establecido a 5, lo que indicará que el valor de dicho campo en la entrada registrada se mantendrá igual, debido a que se está volviendo a registrar dicha entrada.

- **Desunión del nodo y eliminación de la entrada:**

  - En cualquier momento el MAG puede detectar que el nodo móvil se ha desplazado, y por lo tanto ya no está en su link de acceso, o incluso decidir finalizar la sesión de movilidad de un determinado nodo. En ambos casos, el MAG deberá informar al LMA, enviándole un PBU con valor 0 en el campo tiempo de vida.
  - Al igual que anteriormente, en dicho PBU debe existir una opción de prefijo de la *home network* del nodo por cada prefijo asignado a la sesión de movilidad de dicho nodo.
  - El indicador de *handoff* ha de tener un valor de 4, estado desconocido.
  - Una vez que se recibe el mensaje PBA procedente del LMA, cuyo valor en el campo estado es igual a 0, el MAG deberá realizar las siguientes tareas:

    - Borrar la entrada del nodo de su BUL.
    - Borrar la ruta asignada para encaminar el tráfico de dicho nodo.
    - Si existe un túnel bidireccional entre el MAG y el LMA dinámico, y ningún otro nodo va a hacer uso de él, deberá ser eliminado.
    - Deberá derribar el link punto-a-punto que compartía con el nodo.

Cuando el MAG construye un PBU, dentro de la cabecera de movilidad, éste deberá contener los siguientes campos:

- **Dirección origen:** deberá estar presente siempre. Será una dirección configurada en la salida del MAG. En caso de no existir el campo *Proxy-CoA*, esta dirección será considerada como tal.
- **Dirección destino:** deberá estar presente, y además coincidirá con la dirección del LMA.
- **Identificador del nodo:** campo obligatorio.
- **Prefijo de la *home network*:** al menos una instancia de este campo deberá estar presente en el PBU.
- **Identificador de Handoff:** deberá incluirse siempre.
- **Tipo de la tecnología de acceso:** campo obligatorio.
- **Identificador del link de acceso:** puede estar o no presente en el nodo.
- **Sello temporal:** puede ser incluido o no en el PBU.

- **Dirección del link de acceso:** al igual que los dos anteriores su presencia no será obligatoria.

Destacar, que la *Home Address* del nodo, al contrario que en Mobile IPv6, no deberá estar presente en el mensaje construido. Este tipo de mensajes, al igual que los demás deberán ser protegidos con IPsec.

Una vez que un nodo se une a un link, lo primero que ha de conseguir será un *router* por el que encaminar todos los mensajes que quiera enviar. Para ello el nodo deberá enviar una solicitud de *router*. El MAG, cuando recibe dicha solicitud, y antes de contestarla, deberá:

- Siempre que se reciba un mensaje de este tipo se deberá contestar al nodo que lo envió una vez se haya registrado dicho nodo en la BUL, y se haya completado con éxito el registro en el LMA.
- Si el LMA ha rechazado el PBU de registro del nodo, o si el MAG por cualquier motivo no puede completar su regristo, no se le enviará ningún tipo de contestación al nodo.
- Es recomendable que el MAG añada en la respuesta el MTU (*maximum transfer unit*) del túnel bidireccional entre el MAG y el LMA, asegurando así que el nodo use dicha unidad, y aprovechando más eficazmente los recursos disponibles.

Para el nodo móvil, el *router* por defecto siempre será el MAG a cuyo link esté unido en ese momento. Para encaminar los paquetes pertenecientes al nodo, será el MAG el que envíe las solicitudes de *router*. Independientemente de que MAG envíe dicha solicitud, para un mismo nodo, siempre se enviará las mismas direcciones. Para ello, puede que las direcciones a usar por el nodo sean fijas y globales, para que todas las entidades las conozcan y las puedan utilizar, o bien, el MAG puede pedir al LMA que le facilite la configuración del nodo. Ambas posibilidades requieren el descubrimiento de vecinos, y que además éste se haga de una manera segura, como podría ser el uso de clave pública y clave privada.

En PMIPv6 el responsable de reenviar los paquetes perdidos, así como de limitar los PBUs enviados al LMA. Según se especifica en [4], éstas serían las reglas para llevarlo a cabo (conviene recordar que en Mobile IPv6 el que debe realizar estas tareas es el propio nodo móvil):

- Determinar el temporizador de retransmisión, dependiendo del mensaje que se vaya a enviar:
    - Si se solicita prefijos, entonces deberá establecerse a 3 segundos.
    - Si se envía un PBU para registrar el nodo por primera vez, se establecerá a 1.5 segundos.
    - En cualquier otro caso, dicho temporizador será de 1 segundo.

- Se deberá reenviar tantas veces como sea necesario el PBU hasta que se reciba una respuesta del LMA.
- Para dichas retransmisiones se utilizará un *backoff exponencial*, es decir el temporizador de retransmisión se doblará cada vez que se retransmita un paquete hasta un máximo de 32 segundos.
- Para distintos tipos de paquetes, se utilizará diferentes procesos de *backoff*. Se utilizará la misma medida para mensajes de distinto nodos, pero el mismo tipo, lo cual asegurará que a la entidad que reciba los mensajes de ese tipo le dará tiempo a procesar todos antes de volverlos a retransmitir.
- No se enviarán más de 3 mensajes del mismo tipo a la misma entidad.
- Cada retransmisión de un paquete, incrementará en uno el número de secuencia del paquete transmitido anteriormente.

Para su funcionamiento en PMIPv6, habrá que considerar lo siguiente:

- Cuando el MAG envíe un PBU, debería usar como constante el valor de 1 segundo para establecer el temporizador de retransmisiones. Nunca deberá establecerse a un valor mayor de 1.5 segundos.
- El MAG deberá retransmitir dicho PBU hasta que reciba una respuesta válida del LMA, asegurándose antes de cada retransmisión que el nodo móvil sigue unido a su link de acceso.
- Al igual que en Mobile IPv6, se utilizará un mecanismo de *backoff exponencial* para establecer el nuevo temporizador.
- En caso de que el mensaje porte sello temporal, se utilizará el mismo que se empleó en el primer mensaje enviado. En caso de utilizar número de secuencia se procederá del mismo modo que en Mobile IPv6.

Escoger un MTU por parte del LMA, del MAG, y del nodo es de vital importancia, ya que de esta manera se podrán enviar mensajes sin tener que ser fraccionados por ninguna entidad para ser transferidos. Para descubrirlo, todas las entidades podrán utilizar los diferentes mecanismos existentes a tal efecto. El MTU puede sufrir cambios, por que será necesario que se compruebe periódicamente su valor.

El recorrido que deberán seguir los paquetes de este tipo de redes es como se muestra en la [*fig.* 4]. El nodo móvil enviará sus paquetes al MAG, éste al LMA, que los encaminará al nodo destino. En caso de ser destino el nodo móvil seguirá el camino inverso.

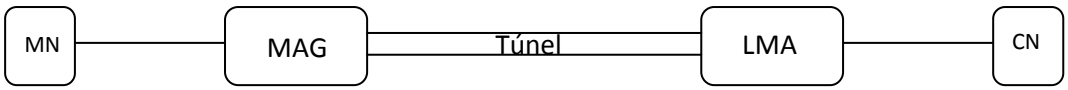


Figura 30: Recorrido mensajes en redes PMIPv6

Dado que el nodo móvil utiliza su *home address* y ésta está ligada el link del MAG en el que se encuentra actualmente, será necesario utilizar las técnicas apropiadas de *tunneling*,

escondiendo de esta forma la topología de red. El MAG y el LMA están unidos mediante un túnel bidireccional, y por tanto los mensajes enviados a través de él deberán ser encapsulados. Pueden ser encapsulados de varias maneras dependiendo el tipo de dirección que tenga el MAG y el LMA (si es IPv6 o IPv4), en este documento, únicamente se tendrá en cuenta cuando ambas direcciones son IPv6.

En caso de que el nodo móvil, y el otro nodo con el que se comunica (CN) se encuentren unidos al mismo link del mismo MAG, el MAG podrá encaminar el tráfico correspondiente con esta comunicación localmente, sin necesidad de enviarla al LMA a través del túnel. La decisión de cómo encaminar este tipo de paquetes dependerá de la política que siga el MAG.

A continuación se exponen las reglas para encaminar los paquetes entre un nodo y el nodo móvil. Existen dos casos posibles:

- **Paquetes con destino el nodo móvil:**
  - Cuando el MAG recibe un paquete procedente del LMA a través del túnel, deberá eliminar las cabeceras correspondientes al enrutamiento del LMA, des encapsularlo y enviar el paquete a la dirección destino que contenga. En caso de error enviará el correspondiente mensaje de ICMP.
  - Si recibe un mensaje de un nodo conectado a su link, para enviarlo a otro nodo que esté también unido a su link, deberá determinar si encaminar el paquete por el túnel o directamente al nodo. Esto se consigue comprobando el valor de un determinado *flag*.

- **Paquetes con origen el nodo móvil:**
  - Cuando el MAG recibe un paquete desde un nodo que está conectado a su link, primero deberá asegurarse que dicho nodo está dado de alta en su BUL, y que el LMA también lo tiene registrado.
  - A continuación deberá comprobar mediante un cierto *flag* si al nodo al que envía el paquete está en el mismo link que el que lo envía. Si coinciden en el mismo link y el MAG está autorizado a enrutarlo directamente, lo hará. De no cumplirse alguna de estas dos condiciones, lo enviará a través del túnel bidireccional al correspondiente LMA.
  - Si el paquete va a ser encaminado por el túnel, se le deberá añadir una cabecera nueva. Dicha cabecera variará en caso de usarse IPsec.

Como se ha comentado con anterioridad se puede utilizar DHCP para configurar un nodo. Para ello:

- DHCP debe ser soportado por todas las entidades MAG del dominio PMIPv6, en cada una de ellas se incluirá una lista de los servidores DHCP que pueden ser utilizados para la configuración.
- Para poder usarlo, la infraestructura de DHCP, también ha de ser configurado para que cree direcciones que estén dentro de unos determinados prefijos. Por lo tanto, para cada link del dominio, la infraestructura de DHCP:

    - Será configurada con una lista de todos los prefijos asociados a ese link.
    - Identificará el link al que se ha unido el nodo móvil por el prefijo que éste utilice.
    - Asignará al nodo una dirección por cada prefijo asociado a dicho link.

- El LMA necesitará conocer todos los prefijos asociados a todos los links del dominio. El LMA asignará todos los prefijos de un determinado link al nodo que se una a dicho link. Como el MAG está físicamente unido a dicho link, podrá realizar la función de agente de DHCP para el nodo.
- Cuando un nodo móvil envíe una solicitud DHCP, el MAG, como su agente deberá establecer el campo dirección del link del mensaje de respuesta con el valor de una dirección que esté contenida dentro de uno de los prefijos de la *home network* del nodo.
- Una vez obtenida la dirección, si el nodo se cambia a otro link, enviará por tanto otra solicitud DHCP, y el nuevo MAG, realizará la misma operación que en el paso anterior.
- Para un correcto funcionamiento del PMIPv6, al nodo siempre se le ha de asignar el mismo conjunto de direcciones sin importar el link en el que se encuentre.

En el caso de que un prefijo deje de ser válido, o sea renombrado, durante una sesión de movilidad el MAG deberá enviar a la dirección del link un aviso de *router* con el tiempo de vida de dicho prefijo establecido a 0. Tanto el MAG como el LMA deberán borrar dicho prefijo así como la ruta establecida para él.

Antes de que el MAG envíe un PBU aumentando el tiempo de vida de una entrada de la *binding cache* del LMA ha de asegurarse que dicho nodo aún sigue unido a su link de acceso. En caso de que no pueda detectar de una manera fiable su unión, no deberá enviar tal PBU. En tal caso, además deberá eliminar la entrada de su BUL, y enviar un PBU con valor 0 en el campo tiempo de vida para que el LMA lo elimine también.

Cuando un nodo entra a formar parte del dominio PMIPv6, el MAG decidirá si se le ofrece o no, servicio de movilidad. Si tiene derecho al servicio, entonces el MAG será el encargado de que dicho nodo no detecte ningún desplazamiento, y lo que conlleva una nueva unión a otro link. En caso de no permitirle la movilidad, será tratado como un nodo ordinario de una red IPv6, por lo que el MAG actuará para él como un simple *router*.

IV. Operaciones del nodo móvil

Cuando un nodo entra a ser parte de un dominio PMIPv6 es porque se ha unido a un determinado link dentro de él. El MAG correspondiente detectará esa unión y se lo comunicará al LMA para llevar a cabo la actualización de ambas entidades. Para ello, típicamente, el nodo deberá haber enviado una solicitud de *router*. El MAG responderá con un mensaje que contenga; los prefijos de las direcciones de la *home network*, una lista de posible *router*s por defecto, y otros parámetros de configuración.

Una vez detectada la aparición de un nuevo nodo en el link de acceso del respectivo MAG, éste iniciará la configuración del nodo y su registro en comunicación con el LMA. Si se recibe una solicitud de *router* por parte del nodo, y no se han completado las operaciones anteriores, el MAG no podrá responder hasta que no finalicen, por lo que se puede detectar cierto retraso.

El nodo podrá ser configurado con DHCP, enviando una solicitud a su agente, o sin DHCP, si el mismo es capaz de auto configurarse. Una vez configurado, el nodo utilizará esa configuración durante todo el tiempo que permanezca en dicho dominio.

Dado que el nodo siempre mantendrá la misma configuración dentro del mismo dominio, siempre detectará el mismo *router*, por lo que no será consciente de su posible cambio de ubicación.

B. Gestión del Proyecto

I. Planificación del proyecto

Como se ha visto en el capítulo de introducción, la realización de este proyecto se puede dividir en cinco grandes fases por las que ha pasado. Cada una de esas fases, a su vez, ha sido dividida en diversas tareas, para así, poder llegar al objetivo final, consiguiendo objetivos intermedios más sencillos de realizar.

Se puede representar la estructura del proyecto mediante el siguiente organigrama:

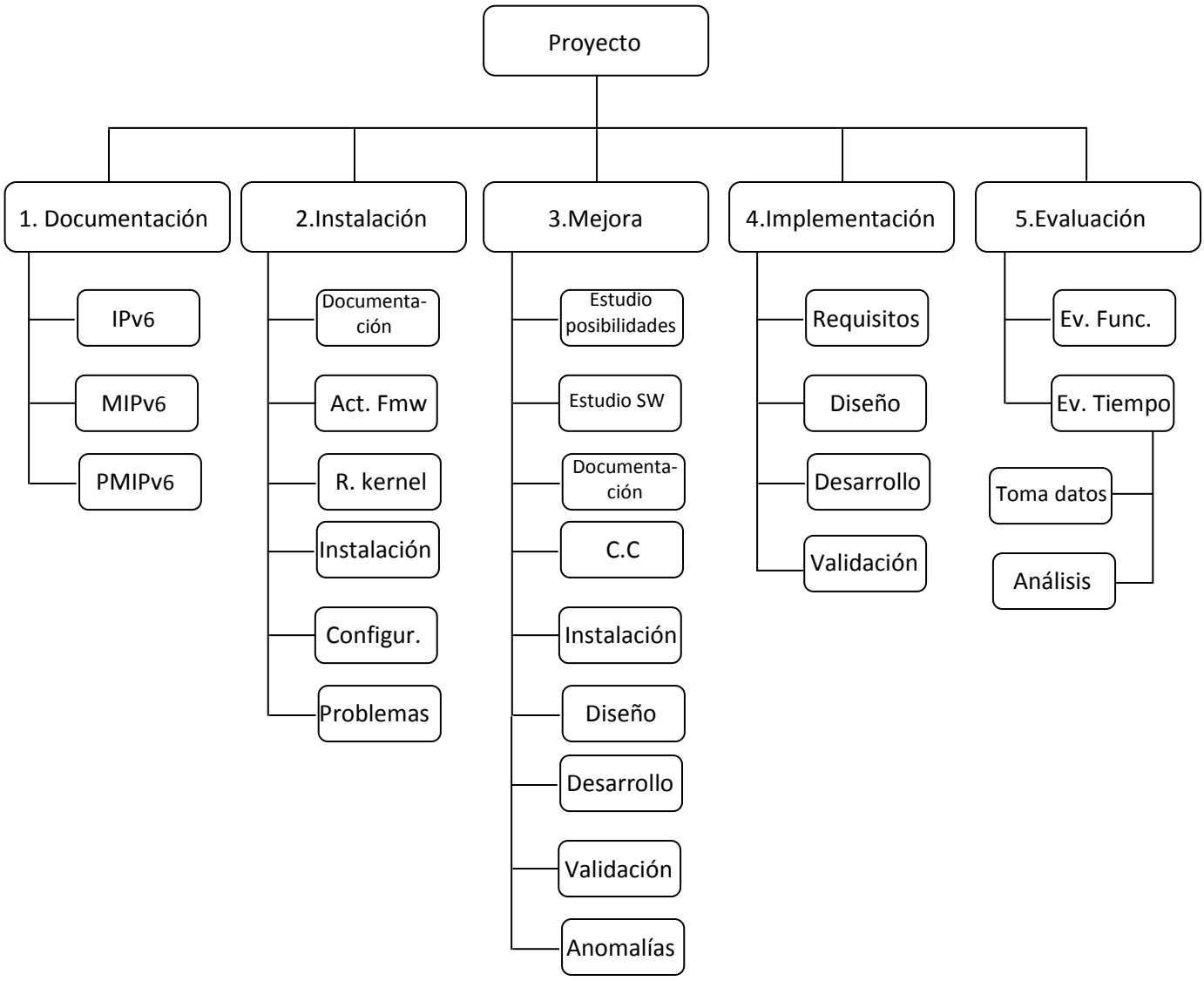


Figura 31: División de tareas

En el organigrama anterior no se han incluido las tareas relativas a la creación del presente documento, ni a los preparativos de la presentación del mismo. Dichas tareas, con sus objetivos, dependencias (ya que ciertas tareas no pueden empezar hasta que hayan terminado otras), duración y esfuerzo son detalladas a continuación:

1. **Documentación**

   - Descripción: estudio de diversos protocolos de movilidad, así como, todos los conceptos en los que están basados.

   - Objetivo: en esta fase se pretende adquirir los conocimientos previos necesarios para poder realizar, a posteriori, el proyecto en sí.

   - Dependencia: no depende de ninguna otra tarea para ser llevada a cabo.

   - Duración: 6 semanas.

   Esta actividad, puede ser dividida en las siguientes tareas:

   a. Estudio general de redes de ordenadores e IPv6:
   - Descripción: estudiar el funcionamiento de una red de ordenadores, así como del protocolo IPv6.
   - Objetivo: adquirir conocimientos sobre IPv6, y sus bases.
   - Dependencia: no depende de ninguna tarea.
   - Duración: 1semana.
   - Recursos: Ingeniero 0.5 hombres/mes.

   b. Estudio de movilidad IPv6 y el protocolo MIPv6:
   - Descripción: entender y comprender el concepto de movilidad IP, y uno de los protocolos que la hacen posible.
   - Objetivo: conocer el problema a resolver, y una de sus posibles soluciones.
   - Dependencia: no puede comenzar hasta que termine la tarea 1.a.
   - Duración: 2 semanas.
   - Recursos: Ingeniero 0.5 hombres/mes.

   c. Estudio del protocolo PMIPv6:
   - Descripción: estudiar el protocolo PMIPv6, en el que se basa este proyecto.
   - Objetivo: aprender el funcionamiento de PMIPv6 en profundidad.
   - Dependencia: no puede comenzar hasta que termine la tarea 1.b.
   - Duración: 3 semanas.
   - Recursos: Ingeniero 0.5 hombres/mes.

2. Instalación primera implementación PMIPv6

- Descripción: Preparación de todos los dispositivos implicados en el entorno de pruebas para que se pueda llevar a cabo las acciones de PMIPv6. A continuación, será necesaria la instalación del programa, así como la configuración de los equipos para que pueda ser ejecutado. Debido a los problemas encontrados, también se añade una tarea de solución de problemas.

- Objetivo: el objetivo principal de esta actividad es conseguir que la primera implementación de PMIPv6 utilizada funcione correctamente, para lo cual también será necesario construir el entorno de pruebas.

- Dependencia: esta tarea no podrá comenzar hasta que no haya concluido la actividad 1.

- Duración: 11 semanas.

Por tanto, las tareas en las que se divide esta fase son:

a. Documentación:
   - Descripción: Estudiar los requisitos que son necesarios para que la implementación de PMIPv6 pueda ser ejecutada correctamente, así como de la metodología a seguir para conseguirlos.
   - Objetivo: conocer qué y cómo se ha de modificar en los equipos con el fin de que se ejecute PMIPv6 en ellos satisfactoriamente.
   - Dependencia: no puede comenzar hasta que termine la tarea 1.c.
   - Duración: 1 semana.
   - Recursos: Ingeniero 0.5 hombres/mes.

b. Actualización *firmware*:
   - Descripción: en la preparación del entorno, el primer paso que se debe realizar será la actualización del *firmware* de los *routers* utilizados en el entorno de pruebas.
   - Objetivo: preparar los *routers* para el entorno de pruebas.
   - Dependencia: no puede comenzar hasta que termine la tarea 2.a.
   - Duración: 1 semana.
   - Recursos: Ingeniero 0.5 hombres/mes.

c. Recompilación de *kernel*:
   - Descripción: en cuanto a los ordenadores de sobremesa se refiere, será necesario activar ciertas características del sistema operativo para que se puedan ejecutar todas las funcionalidades de PMIPv6 correctamente. Sin ellas, no sería posible ciertas acciones.

- Objetivo: recompilación del *kernel* del sistema operativo para activar todas las características del mismo necesarias para la ejecución del programa.
- Dependencia: no puede comenzar hasta que termine la tarea 2.a.
- Duración: 3 semanas.
- Recursos: Ingeniero 0.5 hombres/mes.

d. Instalación de PMIPv6:
   - Descripción: como su propio nombre indica, en esta tarea se llevará a cabo la instalación del programa que implementa el protocolo PMIPv6.
   - Objetivo: instalar PMIPv6.
   - Dependencia: no puede comenzar hasta que terminen las tareas 2.b y 2.c.
   - Duración: 1 semana.
   - Recursos: Ingeniero 0.25 hombres/mes.

e. Configurar equipos:
   - Descripción: una vez instalado el programa, será necesario crear los archivos de configuración, uno por cada ordenador de sobremesa, correspondiente con la funcionalidad que, en ellos, va a ser desarrollada.
   - Objetivo: crear los archivos de configuración necesarios para que se pueda ejecutar el programa.
   - Dependencia: no puede comenzar hasta que termine la tarea 2.d.
   - Duración: 2 semanas.
   - Recursos: Ingeniero 0.5 hombres/mes.

f. Resolver problemas:
   - Descripción: una vez finalizadas con éxito todas las tareas de esta fase, el programa debería funcionar correctamente, pero no es así. Se encuentran una serie de deficiencias y de errores no esperados. Por lo tanto será necesario conocer qué es lo que está fallando, y además corregirlo para que el programa funcione como se espera.
   - Objetivo: investigar y comprender dónde está el error, y depurarlo, para que finalmente, se pueda conseguir que el programa funcione correctamente.
   - Dependencia: no puede comenzar hasta que termine la tarea 2.e.
   - Duración: 3 semanas.
   - Recursos: Ingeniero 1 hombres/mes.

3. Mejora de las funcionalidades

- Descripción: una vez superada la fase anterior, se procede a probar el programa instalado, percibiendo ciertos requisitos necesarios para que su funcionamiento sea el esperado, que no son viables en un entorno real de pruebas. Por lo tanto, se procede a adaptar la implementación, para que su funcionamiento sea el esperado en más circunstancias a lo que lo era inicialmente.

- Objetivo: adaptar la implementación a un entorno de funcionamiento que se asemeje más a la realidad.

- Dependencia: esta tarea no podrá comenzar hasta que no haya concluido la actividad 2.

- Duración: 17 semanas.

Las tareas llevadas a cabo durante el desarrollo de esta fase son:

a. Estudio de las posibilidades:
   - Descripción: estudiar detalladamente las deficiencias del programa, así como de las diferentes posibilidades presentadas para sus mejoras.
   - Objetivo: conocer exhaustivamente, el aumento de funcionalidad propuesto para esta implementación.
   - Dependencia: no puede comenzar hasta que termine la tarea 2.f.
   - Duración: 1 semana.
   - Recursos: Ingeniero 0.25 hombres/mes.

b. Estudio del nuevo SW aportado:
   - Descripción: una vez decidió cual será el aumento de funcionalidad que se va a llevar a cabo, se nos facilita un programa ya implementado para detectar cuando un nodo se ha conectado a un determinado punto de acceso. Es en esta tarea dónde nos familiarizamos con su funcionamiento.
   - Objetivo: conocer el nuevo SW que va a ser incluido tanto en el entorno de pruebas como en el mismo programa.
   - Dependencia: no puede comenzar hasta que termine la tarea 3.a.
   - Duración: 1 semana.
   - Recursos: Ingeniero 0.25 hombres/mes.

c. Documentación:
   - Descripción: para que este nuevo SW pueda ser integrado en nuestro entorno de pruebas, se deben realizar una serie de acciones previas. En esta tarea se estudiaran tanto que tareas son necesarias llevar a cabo, cómo el procedimiento para realizarlas.
   - Objetivo: cuando esta tarea haya finalizado, se deben conocer las tareas que se deben llevar a cabo para la integración del programa, y el método para realizarlas.
   - Dependencia: no puede comenzar hasta que termine la tarea 3.a.
   - Duración: 1 semana.
   - Recursos: Ingeniero 0.25 hombres/mes.

d. *Cross compiling*:
   - Descripción: la tarea más importante para integrar dicho SW en nuestro entorno será la correspondiente con la compilación cruzada. Será en esta tarea dónde sea realizada.
   - Objetivo: generar el archivo ejecutable del SW facilitado para que pueda ser ejecutado en los puntos de acceso del entorno de pruebas.
   - Dependencia: no puede comenzar hasta que termine la tarea 3.c.
   - Duración: 2 semanas.
   - Recursos: Ingeniero 0.5 hombres/mes.

e. Instalación de nuevo SW en los *routers*:
   - Descripción: una vez compilado el programa, será necesario que dicho archivo ejecutable sea copiado e instalado en los *routers* del entorno de pruebas.
   - Objetivo: conseguir que en los *routers* del entorno de pruebas el SW adicional se ejecute correctamente.
   - Dependencia: no puede comenzar hasta que termine la tarea 3.c y 3.d.
   - Duración: 1 semana.
   - Recursos: Ingeniero 0.25 hombres/mes.

f. Diseño:
   - Descripción: una vez que se encuentra preparado de nuevo el entorno de pruebas, se procederá al diseño de la nueva funcionalidad, para que pueda ser integrada en el código fuente original.
   - Objetivo: conocer la manera en la que la nueva funcionalidad va a ser integrada en el programa original.
   - Dependencia: no puede comenzar hasta que termine la tarea 3.e.
   - Duración: 1 semana.
   - Recursos: Ingeniero 0.5 hombres/mes.

g. Desarrollo:
   - Descripción: conocida la manera de integrar la nueva funcionalidad, el siguiente paso será desarrollarla para poder llevarla a cabo.
   - Objetivo: cuando esta fase finalice, se obtendrá como resultado una nueva implementación, en la cual se encuentre la nueva funcionalidad añadida.
   - Dependencia: no puede comenzar hasta que termine la tarea 3.f.
   - Duración: 3 semanas.
   - Recursos: Ingeniero 1 hombres/mes.

h. Validación:
   - Descripción: después de integrar la nueva funcionalidad, se le deberán pasar una serie de pruebas para comprobar que efectivamente se ha realizado correctamente.

- Objetivo: asegurarse de que la nueva funcionalidad se ejecuta de la manera esperada.
- Dependencia: no puede comenzar hasta que termine la tarea 3.g.
- Duración: 1 semana.
- Recursos: Ingeniero 1 hombres/mes.

i. Observación de anomalías:
- Descripción: durante el período de validación del funcionamiento de lo recientemente añadido, se observaron varias desviaciones del mismo. Es en esta tarea dónde dichas desviaciones fueron investigadas.
- Objetivo: conocer el porqué del mal funcionamiento del programa tras añadir una nueva funcionalidad.
- Dependencia: no puede comenzar hasta que termine la tarea 3.h.
- Duración: 6 semanas.
- Recursos: Ingeniero 0.75 hombres/mes.

4. Implementación de PMIPv6

- Descripción: tras el intento fallido del aumento de funcionalidad de la implementación inicial, se decide desarrollar un nuevo programa que implemente lo básico de PMIPv6 en un entorno real, es decir con la nueva funcionalidad que se pretendía añadir anteriormente, y que además se consiga el restablecimiento de la comunicación entre las entidades, siempre en un tiempo aceptable.

- Objetivo: al finalizar esta tarea se habrá conseguido una implementación de PMIPv6 real, y además muy estable.

- Dependencia: esta tarea no podrá comenzar hasta que no haya concluido la actividad 3.

- Duración: 16 semanas.

Esta fase se divide en las siguientes tareas:

a. Obtención de requisitos:
- Descripción: dado que no van a ser implementadas todas las funcionalidades de PMIPv6, será necesario definir cuáles serán las que sí se lleven a cabo y cuáles no.
- Objetivo: a la finalización de esta tarea se sabrá concretamente qué acciones de PMIPv6 serán las necesarias que se lleven a cabo, para esta nueva implementación.
- Dependencia: no puede comenzar hasta que termine la tarea 3.i.
- Duración: 1 semana.
- Recursos: Ingeniero 0.5 hombres/mes.

b. Diseño:
   - - Descripción: descritas las actividades de PMIPv6 que se desean implementar es necesario, en este punto, estudiar la forma de realizarlo.
   - Objetivo: conocer el diseño, es decir de qué manera va a ser realizado el nuevo programa.
   - Dependencia: no puede comenzar hasta que termine la tarea 4.a.
   - Duración: 2 semanas.
   - Recursos: Ingeniero 0.5 hombres/mes.

c. Implementación:
   - Descripción: conocidos los requisitos del programa, y la manera de implementarlos, ahora se escribirá el código fuente del mismo.
   - Objetivo: escribir el código fuente del nuevo programa que lleve a cabo las funcionalidades escogidas de PMIPv6.
   - Dependencia: no puede comenzar hasta que termine la tarea 4.b.
   - Duración: 10 semanas.
   - Recursos: Ingeniero 0.75 hombres/mes.

d. Validación:
   - Descripción: terminada la tarea de implementación será necesario que el programa realiza lo que se quiere que realice, y de la forma que se diseño.
   - Objetivo: asegurarse que el funcionamiento del programa es el correcto.
   - Dependencia: no puede comenzar hasta que termine la tarea 4.c.
   - Duración: 3 semanas.
   - Recursos: Ingeniero 0.75 hombres/mes.

5. Evaluación del funcionamiento

- Descripción: Obtenido el nuevo programa, será necesario hacer un examen más exhaustivo de su funcionamiento. Además, se ha considerado necesario, también, una evaluación del tiempo que es necesario para que todos los nodos del entorno de pruebas vuelvan a ser accesibles por los demás.

- Objetivo: el objetivo de esta última etapa del desarrollo del proyecto consistirá en validar el funcionamiento de la implementación, y además estudiar el tiempo necesario para que se restablezca la comunicación entre los dispositivos.

- Dependencia: esta tarea no podrá comenzar hasta que no haya concluido la actividad 4.

- Duración: 4 semanas.

Las tareas necesarias para concluir esta fase son:

a. Evaluación del funcionamiento:
   - Descripción: durante la realización de esta tarea se comprobará que el funcionamiento del programa es el esperado, y que además se restablece la comunicación en todas las circunstancias probadas.
   - Objetivo: comprobación del restablecimiento de la comunicación, es decir que todos los nodos del entorno de pruebas vuelven a ser accesibles por los demás.
   - Dependencia: no puede comenzar hasta que termine la tarea 4.c. Puede desarrollarse en paralelo con la tarea 4.d.
   - Duración: 1 semana.
   - Recursos: Ingeniero 0.75 hombres/mes.

b. Evaluación del tiempo de incomunicación:
   - Descripción: estudiar detalladamente el tiempo en el que el MN es inaccesible tras cambiar de punto de acceso a la red.
   - Objetivo: obtener un estudio detallado sobre el tiempo que se tarda en restablecer la comunicación entre todos los nodos del entorno de pruebas.
   - Dependencia: no puede comenzar hasta que termine la tarea 5.a.
   - Duración: 3 semanas.

   Además, esta tarea es subdividida a su vez, en otras dos tareas más:

   i. Toma de datos:
      - Descripción: para realizar el estudio estadístico del tiempo para recuperar la comunicación, previamente es necesario obtener datos de prueba. Para que el estudio sea más exhaustivo, dichas medidas del tiempo se tomarán repetidas veces y con parámetros distintos
      - Objetivo: conseguir datos de prueba para su posterior análisis.
      - Dependencia: no puede comenzar hasta que termine la tarea 5.a.
      - Duración: 1 semana.
      - Recursos: Ingeniero 0.5 hombres/mes.

   ii. Análisis estadístico:
      - Descripción: con los datos obtenidos en la tarea anterior, ahora se procede a ser analizados y a obtener una conclusión de ellos.
      - Objetivo: el objetivo de esta tarea es la de dar una estimación lo más ajustada, del tiempo que se necesita para que el MN pueda enviar y recibir mensajes de nuevo tras haber cambiado de punto de acceso a la red.
      - Dependencia: no puede comenzar hasta que termine la tarea 5.b.i.
      - Duración: 2 semanas.
      - Recursos: Ingeniero 0.5 hombres/mes.

Además, se podría considerar otra fase importante en la realización del proyecto consiste en la escritura de la documentación pertinente del mismo.

6. Escritura de la documentación del proyecto

- Descripción: durante el desarrollo de esta etapa se escribirá el presente documento, así como se realizará la presentación del mismo.

- Objetivo: documentar el proyecto realizado. Además, se prepara la presentación del mismo ante el tribunal.

- Dependencia: cada capítulo de este documento no puede ser escrito, hasta que no termine su respectiva fase dentro del proyecto.

- Duración: 3 semanas.

Por tanto, esta actividad, únicamente consta de dos tareas:

a. Escritura del presente documento:
   - Descripción: como su propio nombre indica, en esta tarea se realizará la escritura de la documentación perteneciente al proyecto, es decir, el presente documento.
   - Objetivo: obtener el presente documento.
   - Dependencia: la escritura de cada capítulo deberá sólo podrá ser realizada cuando se haya completado su correspondiente fase en el proyecto.
   - Duración: 2 semanas.
   - Recursos: Ingeniero 1 hombres/mes.

b. Preparación de la presentación del proyecto:
   - Descripción: preparar la presentación del proyecto ante el tribunal.
   - Objetivo: preparar la presentación de este proyecto ante el tribunal.
   - Dependencia: todas las tareas anteriores, han debido ser finalizadas antes de comenzar con ésta.
   - Duración: 1 semana.
   - Recursos: Ingeniero 1 hombres/mes.

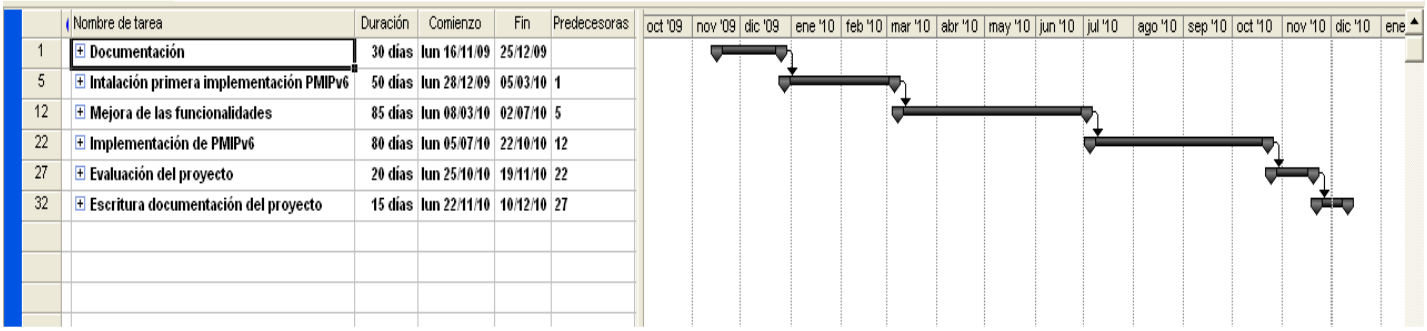
Como medida del tiempo de la duración de todas las tareas, ha sido considerada una semana. Entendiendo por semana únicamente 5 días laborables, es decir de lunes a viernes. En cuanto al cálculo de los recursos humanos utilizados, se ha considerado que una persona trabaja únicamente 8 horas al día. Por lo tanto, si los recursos están medidos en hombres/mes, si el valor es 1, indicará que esa persona ha trabajado 8 horas al día en esa tarea durante un mes entero, es decir 4 semanas de 5 días cada una de ellas.

En la siguiente tabla se muestra un resumen de toda la información sobre la división de actividades y tareas. En ella, además, se especifica tanto el tiempo empleado como los recursos consumidos por cada una de las tareas. Se puede observar que el total de horas invertido es de 1490.

Tarea	Duración (semanas)	Recursos (hombres/mes)	Total horas
1. Documentación			
1.a Estudio general de redes e IPv6	1	0.5	20
2.b Estudio de movilidad y el protocolo MIPv6	2	0.5	40
1.c Estudio del protocolo PMIPv6	3	0.5	60
Total	6		120
2. Instalación de la primera implementación			
2.a Documentación	1	0.5	20
2.b Actualización *firmware*	1	0.5	20
2.c Recompilación *kernel*	3	0.5	60
2.d Instalación PMIPv6	1	0.25	10
2.e Configuración equipos	2	0.5	40
2.f Resolver problemas	3	1	120
Total	11		270
3. Mejora de las funcionalidades			
3.a Estudio de las posibilidades	1	0.25	10
3.b Estudio del nuevo SW aportado	1	0.25	10
3.c Documentación	1	0.25	10
3.d *Cross Compiling*	2	0.5	40
3.e Instalación SW en *routers*	1	0.25	10
3.f Diseño	1	0.5	20
3.g Desarrollo	3	1	120
3.h Validación	1	1	40
3.i Observación anomalías	6	0.75	180
Total	17		440
4. Implementación de PMIPv6			
4.a Obtención de requisitos	1	0.5	20
4.b Diseño	2	0.5	40
4.c Implementación	10	0.75	300
4.d Validación	3	0.75	90
Total	16		450
5. Evaluación del funcionamiento			
5.a Evaluación del funcionamiento	1	0.75	30
5.b Evaluación del tiempo de incomunicación	3		
5.b.i Toma de datos	1	0.5	20
5.b.ii Análisis estadístico	2	0.5	40
Total	4		90
6. *Escritura de la documentación del proyecto*			
6.a Escritura del presente documento	2	1	80
6.b Preparación de la presentación del proyecto	1	1	40
Total	3		120
			1490

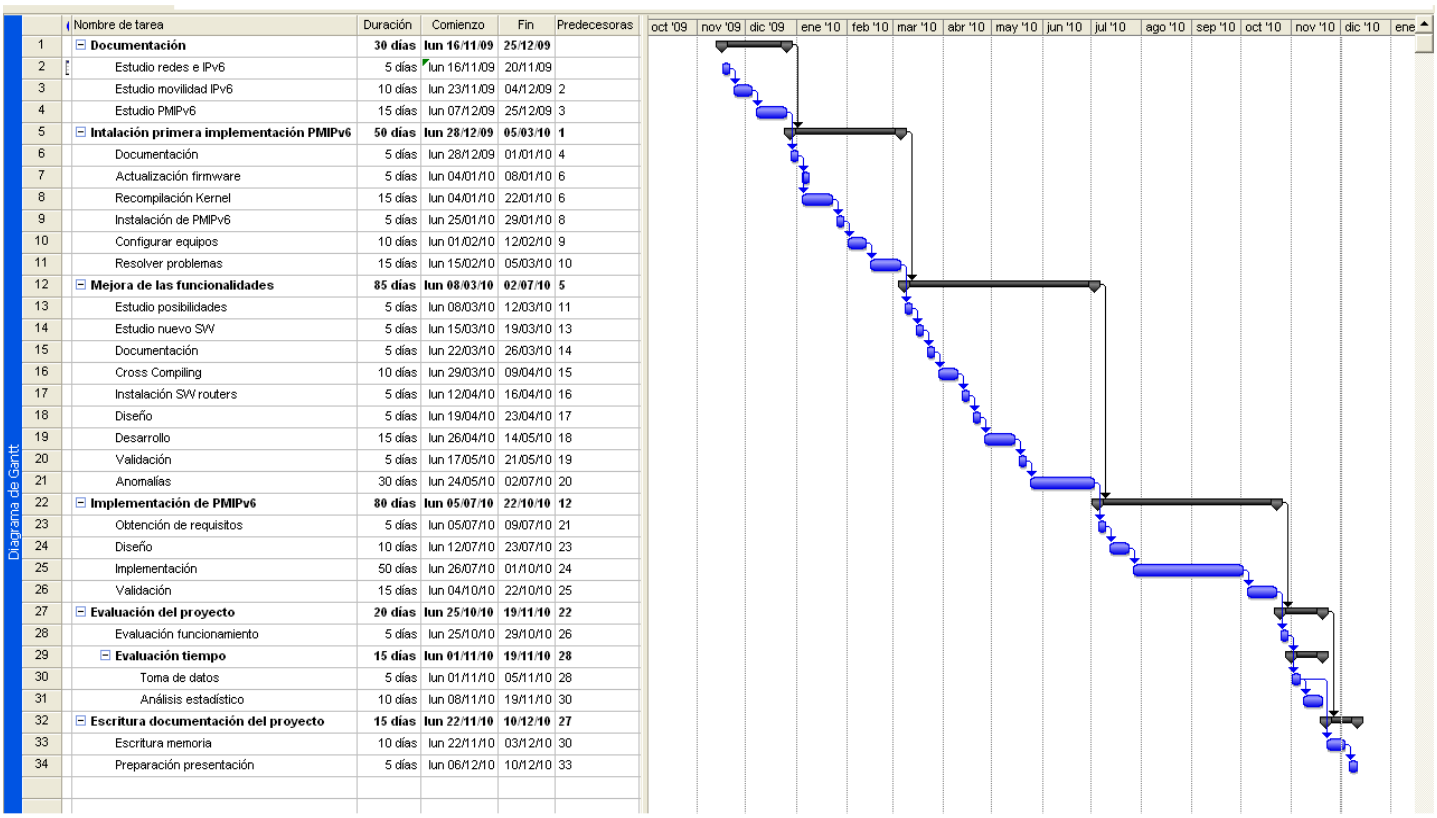
Figura 32: Resumen de tareas y reparto de horas

La *fig.* 32 corresponde con el diagrama Gantt, de las fases generales de la elaboración del proyecto. En la figura siguiente a ella, la *fig.* 33, el diagrama Gantt mostrado corresponde con el detalle de cada una de esas fases, en el que se incluyen las tareas realizadas.



	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	Documentación	30 días	lun 16/11/09	25/12/09	
5	Intalación primera implementación PMIPv6	50 días	lun 28/12/09	05/03/10	1
12	Mejora de las funcionalidades	85 días	lun 08/03/10	02/07/10	5
22	Implementación de PMIPv6	80 días	lun 05/07/10	22/10/10	12
27	Evaluación del proyecto	20 días	lun 25/10/10	19/11/10	22
32	Escritura documentación del proyecto	15 días	lun 22/11/10	10/12/10	27

Figura 33: Diagrama Gantt reducido



	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	**Documentación**	**30 días**	**lun 16/11/09**	**25/12/09**	
2	Estudio redes e IPv6	5 días	lun 16/11/09	20/11/09	
3	Estudio movilidad IPv6	10 días	lun 23/11/09	04/12/09	2
4	Estudio PMIPv6	15 días	lun 07/12/09	25/12/09	3
5	**Intalación primera implementación PMIPv6**	**50 días**	**lun 28/12/09**	**05/03/10**	**1**
6	Documentación	5 días	lun 28/12/09	01/01/10	4
7	Actualización firmware	5 días	lun 04/01/10	08/01/10	6
8	Recompilación Kernel	15 días	lun 04/01/10	22/01/10	6
9	Instalación de PMIPv6	5 días	lun 25/01/10	29/01/10	8
10	Configurar equipos	10 días	lun 01/02/10	12/02/10	9
11	Resolver problemas	15 días	lun 15/02/10	05/03/10	10
12	**Mejora de las funcionalidades**	**85 días**	**lun 08/03/10**	**02/07/10**	**5**
13	Estudio posibilidades	5 días	lun 08/03/10	12/03/10	11
14	Estudio nuevo SW	5 días	lun 15/03/10	19/03/10	13
15	Documentación	5 días	lun 22/03/10	26/03/10	14
16	Cross Compiling	10 días	lun 29/03/10	09/04/10	15
17	Instalación SW routers	5 días	lun 12/04/10	16/04/10	16
18	Diseño	5 días	lun 19/04/10	23/04/10	17
19	Desarrollo	15 días	lun 26/04/10	14/05/10	18
20	Validación	5 días	lun 17/05/10	21/05/10	19
21	Anomalías	30 días	lun 24/05/10	02/07/10	20
22	**Implementación de PMIPv6**	**80 días**	**lun 05/07/10**	**22/10/10**	**12**
23	Obtención de requisitos	5 días	lun 05/07/10	09/07/10	21
24	Diseño	10 días	lun 12/07/10	23/07/10	23
25	Implementación	50 días	lun 26/07/10	01/10/10	24
26	Validación	15 días	lun 04/10/10	22/10/10	25
27	**Evaluación del proyecto**	**20 días**	**lun 25/10/10**	**19/11/10**	**22**
28	Evaluación funcionamiento	5 días	lun 25/10/10	29/10/10	26
29	**Evaluación tiempo**	**15 días**	**lun 01/11/10**	**19/11/10**	**28**
30	Toma de datos	5 días	lun 01/11/10	05/11/10	28
31	Análisis estadístico	10 días	lun 08/11/10	19/11/10	30
32	**Escritura documentación del proyecto**	**15 días**	**lun 22/11/10**	**10/12/10**	**27**
33	Escritura memoria	10 días	lun 22/11/10	03/12/10	30
34	Preparación presentación	5 días	lun 06/12/10	10/12/10	33

Figura 34: Diagrama Gantt

En ambas figuras se puede observar la dependencia existente entre las tareas realizadas.

II. Recursos empleados

El desarrollo de este proyecto ha sido realizado íntegro en el laboratorio 41F04 del departamento de Ingeniería Telemática. En él se disponía de los siguientes dispositivos:

- 3 ordenadores de sobremesa Intel Core Quad , 2.66 GHz, 3 GB de memoria RAM cuyo sistema operativo es Linux (distribución Ubuntu).
- Un ordenador portátil (actúa como MN en el entorno de pruebas) Intel Core Duo, 1.20 GHz, 2 GB de memoria RAM, con dos tarjetas de red inalámbricas, una insertada en él y otra externa, marca NETGEAR. Su sistema operativo coincide con el de los otros tres ordenadores.
- 2 *routers* inalámbricos de la marca Linksys modelo WRT54GL.
- Un *switch* para poder conectar los tres ordenadores de sobremesa.
- Diversos cables de red para conectar todos los elementos del entorno de pruebas descrito en el apartado 5.1.

- La realización del presente documento, se ha llevado a cabo en otro ordenador con sistema operativo Windows Vista.

En cuanto a SW se refiere, han sido utilizados los siguientes:

- Sistema Operativo en todos los ordenadores disponibles, Linux, distribución Ubuntu 8.9.
- El *firmware* escogido en los *routers* corresponde con OpenWRT/ kamikaze/ 8.09.2/BRCM-2.4.
- El *sniffer* escogido para capturar los mensajes enviados a través de la red por los ordenadores de sobremesa del entorno de pruebas, y poder de este modo ser analizados posteriormente, es *Wireshark*.
- En cuanto al ordenador portátil se refiere, para controlar los mensajes tanto enviados como recibidos, el *sniffer* utilizado corresponde con *tcpdump*.
- Dado que ambas implementaciones han sido desarrolladas en C, el compilador en este caso coincidirá con *gcc*.
- Para escribir el código fuente ha sido utilizado un editor de texto incluido en el sistema operativo, denominado *Gedit*.
- Este documento ha sido escrito gracias a Microsoft Office 2007, en concreto a Microsoft Word.
- Además ha sido utilizado el programa *PDFCreator* para modificar el formato del texto.

III. Presupuesto

Para calcular el coste estimado del proyecto se han tenido en cuenta, tanto los costes acarreados por los recursos humanos, como los costes devenidos por los recursos materiales.

En primer lugar, se calculará el coste asociado a las herramientas SW empleadas durante el desarrollo. Este coste es debido al pago de diversas licencias para poder utilizar determinados programas.

HERRAMIENTA SW	COSTE LICENCIA (€)
Ubuntu 8.9	LIBRE
Windows Vista Home Premium	181.18
Firmware *Routers*	LIBRE
Wireshark	LIBRE
TCPDump	LIBRE
Gedit	LIBRE
Microsoft Office 2007	470.94
PDFCreator	LIBRE
TOTAL (€)	652,12

Tabla 5: Costes Sw

A continuación, se calculará el coste producido por los recursos humanos. Como se ha mostrado en esta misma sección el total de horas empleado en la realización del proyecto es de 1490. Por tanto:

RECURSO HUMANO	TOTAL HORAS	€/HORA
Ingeniero en Informática	1490	35
TOTAL (€)		52.150

Tabla 6: Costes recursos humanos

En cuanto a los recursos materiales se refiere, los gastos acarreados por los cables de red y el hub para poder crear el escenario de pruebas, se han considerado despreciables, dado su bajo coste. Por lo que, la estimación del coste, teniendo en cuenta el período de tiempo en el que los recursos han sido utilizados, es la siguiente:

RECURSO MATERIAL	CANTIDAD	PRECIO UNITARIO (€)	% DEDICADO	MESES DEDICACIÓN	PERIODO DEPRECIACIÓN (meses)	TOTAL (€)
Ordenador Sobremesa	3	550	100	13	60	357.5
Ordenador Portátil	1	700	100	13	60	151.66
Tarjeta de red inalámbrica	1	49	100	13	60	10.61
Router Linksys	2	60	100	13	60	26
Switch D-Link	1	12	100	13	60	2.6
Equipamiento de red	-	-	-			
TOTAL (€)						548.37

Tabla 7: Costes recursos materiales

El cálculo del coste imputable de los recursos materiales, ha sido calculado con respecto a la siguiente fórmula:

$$\frac{a}{b} * c * d$$

Dónde:

a: número de meses que el equipo ha sido utilizado.
b: periodo de depreciación.
c: coste del equipo sin IVA.
d: porcentaje de uso del equipo dedicado al proyecto.

Los costes indirectos que el proyecto haya podido acarrear, como puede ser el consumo de energía eléctrica, entre otros, han sido estimados en un 20% del total del proyecto.

El coste total del desarrollo del proyecto, será calculado mediante la suma del coste de las herramientas SW, el coste de los recursos humanos, el coste de los recursos materiales, y los costes indirectos derivados de la realización del proyecto. Es por tanto:

CONCEPTO	TOTAL (€)
Herramientas SW	652.12
Recursos Humanos	52150
Amortización	548.37
Costes indirectos	10670.1
TOTAL (€)	64.020,59

Tabla 8: Costes totales

El coste total del desarrollo del proyecto asciende a la cantidad de **64.020,59 € (sesenta y cuatro mil veinte euros con cincuenta y nueve céntimos de euro).**

C. Preparación del entorno de pruebas

I. Preparación de *routers*

Los puntos de accesos son *routers* wireless Linksys WRT54GL. El primer paso para construir este entorno, será configurar los *routers* mencionados. Para ello, se les instalará otro *firmware* que nos permitirá en un futuro añadirle funcionalidades a los *routers*. El *firmware* escogido es: OpenWRT/kamikaze/8.09.2/BRCM-2.4. Es un software libre, por lo que accederemos directamente a la página web [11], y en *downloads* seleccionar el *firmware* mencionado. Una vez que se accede a la lista de posibles *firmwares*, se procede a descargarse el archivo correspondiente a nuestro modelo de *routers*. El archivo descargado será por tanto: Openwrt-wrt54g-squashfs.bin.

Una vez descargado este archivo, y para instalar el nuevo *firmware* en el *router*, éste se conecta a un ordenador. Se inicia un navegador web y en la barra de direcciones se escribe la dirección IP por defecto que tiene el *router*: 192.168.1.1. De esta forma se inicia la interfaz web del *router*. Para acceder a ella, es necesario identificarse con nombre de usuario y contraseña. Ambos campos inicialmente será "admin". A continuación, accedemos a *Firmware upgrade* que se encuentra dentro de la pestaña de *Administración*. Ahí, nos encontraremos un espacio para escoger el archivo que queremos cargar en el *router*. Destacar que dicho archivo deberá tener la extensión ".bin". Escogido el archivo correcto, procedemos a cargarlo en el *router* pulsando el botón *Upgrade.* La duración de este proceso es de varios minutos, en los cuales es muy importante no desconectar el *router* del ordenador, ya que éste puede quedar inservible.

Al realizar este cambio de *firmware*, para acceder ahora al *router* como administrador el nombre de usuario será "root" y la contraseña "root".

Otro aspecto necesario para el correcto funcionamiento de nuestro escenario es la configuración de la red WIFI generada por ambos puntos de acceso. Es recomendable que la red de ambos *routers* se llame exactamente igual, ya que en un entorno real en el protocolo PMIPv6 el nodo móvil cambia de punto de acceso dentro de la misma red por diferentes motivos. Para ello, accedemos de la misma manera que antes al *router*, pero con el nuevo usuario y la nueva contraseña. Entramos en la pestaña WIFI dónde aparecen diferentes campos con diferentes opciones a elegir, asignamos:

- Modo: AP (Access Point).
- ESSID: definirá el nombre de la red. En ambos *routers* deberá ser el mismo. En este caso se ha llamado "pfc_ana" en el *router* conectado al MAG-1, y "pfc_ana_b" al conectado al MAG-2.

- Channel: definirá el canal en el que se emitirá la señal y por lo tanto su frecuencia. Cada *router* debe emitir por un canal distinto. Por ello a uno se le asignará el canal 3 y al otro el 13.

Cuando se ha configurado las dos redes WIFI, ya se podrá conectarse a cada una de ellas desde el nodo móvil. Para diferenciar el *router* por el cual nos estamos conectando a la red "pfc_ana" o "pfc_ana_b" del entorno, será necesario saber cuál es la dirección MAC de cada WIFI de los *routers*. Así:

	ESSID	Canal	Dirección MAC
***Router*-1**	pfc_ana	8	00:1C:10:A4:2C:50
***Router*-2**	pfc_ana_b	4	00:1C:10:9A:70:65

Tabla 9: Datos *routers* entorno de pruebas

Por lo tanto, para conectarse a estas redes, desde el nodo móvil se deberán tener tanto el nombre de la red y el canal como la dirección MAC. El comando para conectarse está detallado en el anexo "Guía de usuario", ya que además de esta información es necesario información propia del nodo móvil.

Además, será necesario en ambos *routers* eliminar el *firewall,* de este modo se evita que se hagan comprobaciones no necesarias para nuestro entorno de pruebas, ya que es completamente seguro, y de esta forma reducir el tiempo empleado en la realización de todas las operaciones necesarias relativas a PMIPv6. Para ello, será necesario, primeramente conectarse mediante SSH al *router* como usuario "root"y a continuación parar el *firewall* mediante el comando:

#/etc/init.d/firewall stop

Una vez, realizada esta operación, ya podrá ser eliminado, utilizando el comando:

#opkg remove firewall

II. Preparación de MAGs

El siguiente paso, consistirá en configurar los ordenadores que actúan como MAG, para que se comporten de acuerdo a lo esperado. Para el PMIPv6 funcione correctamente, todos los *links* de acceso a nuestro entorno, y por lo tanto al LMA deberán disponer de la misma dirección. Por ello, habrá que modificar la dirección MAC (la habilitada por el fabricante) de una interfaz "eth0_rename" perteneciente a un MAG, y asignarle la misma dirección de dicha interfaz en el otro MAG. La escogida es la perteneciente al MAG-2, por lo que a ella le será asignada la dirección de la interfaz "eth0_rename" del MAG-1. Para ello, con permisos de *root*, en el ordenador que actúa como MAG-2, se ejecuta:

#ip link set eth0_rename address 00:27:19:b0:01:de

Dónde “00:27:19:b0:01:de” es la dirección HW de la interfaz “eth0_rename” perteneciente al MAG-1.

A estas interfaces no se les deberá asignar ninguna dirección global de IPv6, ya que serán accesibles por sus direcciones locales. Debido al método de formación de este tipo de direcciones en IPv6 [2], ambas interfaces compartirán la misma dirección. Dicha dirección será inequívoca en cada uno de los enlaces. Esta dirección es; fe80::227:19ff:feb0:1de/64.

Cómo se menciona en el anexo G, estos ordenadores tiene la opción de *forwarding* activada. Previamente habilitada durante la instalación dado que por defecto se encuentra desactivada.

Por último, en estos ordenadores habrá que asignar direcciones IPv6 a las otras interfaces que intervengan en la ejecución del programa. Para ello se deben configurar las tarjetas de red manualmente. En este caso se modificará la interfaz denominada como “eth1” de los MAGs. Para conseguirlo, se ha de modificar el fichero “interfaces” ubicado en el directorio “/etc/network/”. Una vez abierto, se añade, en el caso del MAG-1:

```
auto eth1
allow-hotplug eth1
iface eth1 inet6 static
        address 2001::3
              netmask 64
```

En el caso de MAG-2, lo añadido sería exactamente igual a diferencia de la dirección de acceso global:

```
auto eth1
allow-hotplug eth1
iface eth1 inet6 static
        address 2001::2
              netmask 64
```

Una vez modificado el archivo, se procede a levantar la interfaz que acabamos de configurar:

#ifconfig eth1 up

Para finalizar, será necesario reiniciar las conexiones para que los cambios surjan efecto. Se ejecuta por tanto:

#/etc/init.d/networking restart

Según se vaya avanzando en la realización del proyecto será necesaria la configuración de la otra interfaz de la que disponen cada uno de los ordenadores que actúan como MAG, con

el fin de poderse conectar a los *routers* a través de SSH. Los detalles sobre los pasos llevados a cabo se pueden encontrar en el Anexo E.

III. Preparación del LMA

En el ordenador que actúa de LMA en nuestro entorno de pruebas, y al igual que en los dos que actúan de MAGs, habrá que configurar manualmente la dirección de IPv6 de la interfaz que le une a los MAGs. Por lo que habrá que modificar el mismo archivo que en el caso anterior, añadiendo:

```
auto eth2
allow-hotplug eth2
iface eth2 inet6 static
        address 2001::1
              netmask 64
```

A continuación levantar la interfaz, al igual que antes, y por último reiniciar las conexiones de la misma manera que se realizó en los MAGs.

Entre el LMA y los dos MAGs se encuentra un *switch* que permite tener conectado a los tres ordenadores.

IV. Preparación del Nodo Móvil

En el nodo móvil, para el correcto funcionamiento del protocolo PMIPv6 no es necesario que se ejecute ningún software, así como tampoco ningún tipo de configuración, ya que no se le deben asignar direcciones específicas de IPv6, ya que deberán ser configuradas por él mismo con el prefijo que le haya sido asignado.

Para el correcto funcionamiento del protocolo en este nodo, deberemos asegurarnos que su opción de "*forwarding*" está desactivada. En el caso de encontrarse desactivada (por defecto), se encontrará activada la opción "*accept_ra*", la cual permitirá al nodo recibir RA además de auto-configurarse con los datos recibidos en él. Al encontrarse activada la opción "*enable_ra*", se encuentra activada la opción "*autoconf*", que será la que posibilite al nodo móvil configurar su dirección a partir de la información del prefijo recibido en el RA.

Para comprobar que dichas opciones están activadas (es decir tienen valor 1), deberemos acceder al directorio "/proc/sys/net/ipv6/conf/" en el cual nos encontraremos todas las interfaces de red de las que dispone nuestro ordenador, o bien el directorio /all/, en el que los cambios serán aplicados a todas las interfaces. Una vez en dicho directorio, se

podrán encontrar todas las opciones posibles, entre las que se encuentran las mencionadas. Será suficiente con abrir cada opción y comprobar que su valor es 1.

D. Recompilación de Kernel

El primer paso a ejecutar en los equipos que actúen como los mencionados roles será la actualización del *kernel.* Para su realización se ha seguido una combinación de los manuales [7][8]. Tal y como se menciona en [7], primeramente será necesario instalar los siguientes paquetes Debian [9]:

- **Build-essential:** paquete necesitado para poder crear y/o compilar otros paquetes. Contiene una lista informativa de los paquetes fundamentales, necesarios para llevar a cabo dichas operaciones.
- **Ncurses*:** la utilidad de estos paquetes es incorporar una funcionalidad básica para crear diferentes casos para verificar el funcionamiento de un programa. Para ello crea; estructuras para los casos de prueba, diferentes interfaces de prueba, así como informar de los resultados.
- **Kernel-package:** utilizado para crear paquetes relacionados con el *kernel*, como pueden ser el paquete *kernel-image* (que posteriormente será el paquete utilizado para la instalación del nuevo *kernel*). Además, provee la funcionalidad de empaquetar las cabeceras del núcleo, así como la automatización del proceso de compilación de él.

Para ello, se ejecuta el comando:

#apt-get install build-essential ncurses* kernel-package

Una vez instalados los paquetes, habrá que obtener las fuentes de nuestro *kernel* actual (2.6.31). Para ello, se ejecuta el comando:

#apt-get install Linux-source

Conseguidas las fuentes, se procede a descomprimirlas:

#tar xvjf linux-source-2.6.31.tar.bz2

A continuación, se crea un enlace al directorio donde se encontrará el nuevo *kernel*.

#ln –s Linux-source-2.6.31 linux

Una vez se haya accedido al directorio en el que se encuentran las fuentes, se realizará una limpieza del árbol del *kernel* para asegurar una compilación en limpio. Para ello, se utiliza el comando:

#make mrproper

El siguiente paso será configurar las opciones que se desea que estén en el nuevo *kernel*. Para ello, se parte del archivo de configuración anterior, ubicado en *boot/config-2.6.31-20-generic-pae*. Por lo tanto, será necesario copiar dicho archivo al directorio donde se encuentran nuestras fuentes y nombrarlo como "*.config"*, tomándolo así como base del nuevo *kernel*. Se inicia el menú de configuración para establecer los parámetros necesarios para el funcionamiento del programa, ejecutando:

#make menuconfig

Una vez abierto, se navega a través de él buscando las opciones que han de tomar el valor "yes", según indica la documentación del programa. Éstas son:

- **CONFIG_EXPERIMENTAL:** proporciona estabilidad y fiabilidad al sistema cuando un programa nuevo está siendo probado.
- **CONFIG_SYSVIPC:** "inter process comunication", este parámetro permite a los procesos (programas) sincronizarse y poder intercambiar información entre ellos.
- **CONFIG_PROC_FS:** con esta opción se consigue tener un sistema de ficheros virtuales, que en cualquier momento puedan ser consultados y proporcionen información útil sobre el estado del sistema.
- **CONFIG_NET:** aporta al sistema el soporte necesario para cualquier tipo de red.
- **CONFIG_INET:** activa todos los protocolos relativos a Internet y a muchos de Ethernet.
- **CONFIG_IPV6:** activa la implementación del protocolo IPv6.
- **CONFIG_IPV6_MIP6:** activa la funcionalidad de movilidad para el protocolo IPv6, tal y como está descrito en [4].
- **CONFIG_XFRM:** es necesitada para que algunas características puedan ser activadas, como por ejemplo IPV6_TUNNEL.
- **CONFIG_XFRM_USER:** cuando esta opción es activada, se da soporte para la configuración de la interfaz de usuario de IPsec.
- **CONFIG_XFRM_SUB_POLICY:** permite que dos políticas sean aplicadas al mismo tiempo a un mismo paquete.
- **CONFIG_INET6_XFRM_MODE_ROUTEOPTIMIZATION:** esta opción da soporte a la optimización del *router* en MIPv6.
- **CONFIG_IPV6_TUNNEL:** activa la opción de encapsular paquetes IPv6 en paquetes IPv6.
- **CONFIG_IP_ADVANCED_*ROUTER*:** activa el soporte para que el sistema pueda actuar como un *router* IPv6.
- **CONFIG_IPV6_MULTIPLES_TABLES:** permite el soporte a diferentes tablas que contengan información sobre IPv6.
- **CONFIG_IPV6_SUBTREES:** activa la posibilidad de encaminar un paquete dependiendo de su dirección origen o de un prefijo.

- **CONFIG_ARPD:** permite el uso del protocolo ARP para resolver direcciones hardware.
- **CONFIG_INET6_ESP:** activa el soporte a IPsec en IPv6.
- **CONFIG_NET_KEY:** activa el cifrado de red. Es imprescindible cuando se quiere utilizar herramientas de IPsec.
- **CONFIG_NET_KEY_MIGRATE:** permite el cambio de claves cifradas en red. Es utilizado para actualizar dinámicamente el localizador de una asociación de seguridad IPsec.

Antes de proceder a la compilación y posterior instalación de nuestro nuevo *kernel*, es recomendable comprobar que la configuración del *kernel* es la que debe ser. Para ello, se facilita un *script* junto al programa, y nos informará, en el caso de que los haya, que parámetros se encuentran mal configurados. Para un funcionamiento adecuado, será necesario pasarle el *path* dónde se encuentra el fichero de configuración.

 # ./chkconf_kernel.sh /usr/src/linux/

Una vez generado y guardado el nuevo fichero de configuración de nuestro *kernel,* éste deberá ser compilado. Para realizar esta tarea de forma más rápida, se debe conocer el número de núcleos de los que dispone el equipo en cat/proc/cpuinfo. El número de núcleos en el equipo en el que se va a compilar el *kernel* es de cuatro, por lo que a continuación se ejecuta:

#export_CONCURRENCY_LEVEL=4

Hecho todo lo anterior, se procede a compilar en nuevo *kernel*.

#make-kpkg buildpackage –initrd –rev pmip.1 kernel.image

Gracias a la opción "-initrd", se creará un paquete con la imagen del *kernel*, que posteriormente será instalado. Además se generarán dos paquetes más; uno con las cabeceras del *kernel* y otro con la documentación, y se encontrarán en directorio usr/src en una carpeta con el mismo nombre que el nombre de nuestra revisión. Estos tres paquetes se copiarán al resto de ordenadores, para no repetir la compilación en ellos, ya que es un proceso que consume mucho tiempo. Esto es posible realizarlo ya que el *kernel* que se desea instalar en todos los equipos es el mismo, y además la distribución de Linux instalada en ellos también coincide.

El último paso para completar la actualización del *kernel*, será su instalación. Para ello será necesario instalar la imagen, las cabeceras y la documentación. Esto se repetirá en los tres equipos en los que se quiere ejecutar este nuevo *kernel*.

#dpkg -i Linux-image-2.6.26.2_2.6.26.2-10.00.Custom_i386.deb
#dpkg -i Linux-headers-2.6.26.2_2.6.26.2-10.00.Custom_i386.deb
#dpkg -i Linux-doc-2.6.26.2_2.6.26.2-10.00.Custom_i386.deb

Por último se reinician todos los ordenadores, y se escoge que se ejecute el *kernel* recién instalado.

Destacar que la secuencia de comandos ha sido siempre realizada como *root.*

E. Cross Compiling

Una vez comprobado el funcionamiento básico del programa, el siguiente paso en el proyecto será ampliar su funcionalidad. Como se ha comentado anteriormente, la implementación inicial del protocolo PMIPv6, únicamente reacciona, es decir, emula el comportamiento del dicho protocolo, si la interfaz del MN, por la cual se conecta a la red del MAG, es levantada en ese momento, con lo cual es la primera vez que se utiliza, y por lo tanto, es cuando se envía el mensaje RS que recibe el MAG (es el único tipo de mensaje que recibe con el cual procede a la ejecución de PMIPv6), y es entonces cuando este comienza a simular el protocolo. Si por el contrario, únicamente nos conectamos a la red del MAG, el MN no envía un RS, y por lo tanto, aunque estemos conectados a la red de un MAG éste no llevará a cabo ninguna acción relativa al protocolo, como puede ser la creación del túnel para comunicarse con el LMA o la creación de la ruta para acceder al MN o incluso la creación de una entrada correspondiente a ese nodo en su BUL, y no se podrá conseguir el objetivo de dicho protocolo, y por tanto el MN no podrá cambiar de ubicación y ser accesible por el resto de nodos de la red.

Por ello, se consideró necesario ampliar la funcionalidad de la implementación inicial, de tal modo que el MAG fuese capaz de detectar cuando un nodo se conecta a su red, y entonces decidir si le proporciona movilidad o no. Por tanto, el funcionamiento de esta nueva implementación, deberá ser igual que el descrito en el apartado 7.F.III, pero en vez de ser estrictamente un RS el mensaje enviado por el MN al MAG, será un mensaje enviado del *router* al MAG indicando que se ha conectado a él un nuevo nodo, y le proporcionará la dirección HW del mismo, que recordemos, es el identificador del MN en el perfil que almacenan tanto los MAG como el LMA, y con ella el MAG, en caso de que el nodo esté autorizado para una posible movilidad, emprenderá las acciones necesarias para poder facilitársela.

Para poder incluir la nueva funcionalidad se necesitará un programa adicional, aunque en realidad, se podría considerar como dos. Uno de ellos será ejecutado en los *routers* que están conectados a los MAGs, que será el encargado de detectar cuando un nuevo nodo se ha conectado a su red, y entonces enviarle un mensaje al MAG comunicándoselo. El otro programa se ejecutará en cada uno de los MAG, y simplemente se quedará a la espera de recibir un mensaje del *router*, que indique si se ha conectado un nuevo nodo, o si por el contrario se ha desconectado. Una vez que dichos programas se ejecuten correctamente, se procederá a su integración en la implementación original.

Ambos programas han sido desarrollados en C, como la aplicación inicial. Será imprescindible compilar dichos programas e instalarnos en los nodos de la red para que puedan funcionar. En los MAG no existe ningún tipo de problema, ya que al ser ordenadores, se podrán realizar dichas operaciones de forma ordinaria. El problema surge cuando se debe compilar e instalar el programa en el *router*. Por ello es estrictamente necesaria la utilización

de *cross compiling*, ya que, aunque el sistema operativo instalado en los *routers* es *Linux*, pero dispone de tan poca memoria que estas operaciones no pueden llevarse a cabo en él.

Se puede definir *Cross Compiling* como el procedimiento llevado a cabo para poder crear ficheros ejecutables para una arquitectura distinta a la arquitectura en la cual se está compilando dicho programa, y crear de este modo el fichero ejecutable. Esta técnica es necesaria aplicarla ya que en el caso que nos ocupa, el programa que será instalado en el *router* será compilado en un ordenador, en el cual si sea posible compilar, y por tanto dichas arquitecturas serán distintas.

Lo primero que será necesario para llevar a cabo este proceso será conocer la arquitectura del dispositivo en el cual se va a ejecutar posteriormente el programa. En este caso, el dispositivo es un *router* de la marca Linksys modelo WRT54GL, al cual como se ha comentado anteriormente en el Anexo C, se le actualizó el *firmware*, y se instaló OpenWRT/kamikaze/8.09.2/BRCM-2.4. Por tanto, la arquitectura de nuestro sistema es BRC-2.4.

Una vez conocido esto, en el ordenador dónde va a ser compilado el código que será ejecutado en los *routers*, será necesario instalar los paquetes:

- Subversion: este paquete permite el control de versiones. Fue diseñado para reemplazar a CVS y corregir alguna de sus deficiencias, aunque en la actualidad aún sigue siendo el más empleado.
- Gawk: utilizado para procesar datos basados en texto, ya sean, o bien en ficheros o bien en flujos de datos. Fue necesario instalarlo, dado que llegado a un punto de la *cross compiling*, existían dependencias y se sugería que se instalase este paquete para evitarlas.

Para ello, y como si hizo anteriormente para instalar paquetes de Linux en los ordenadores, se ejecutarán los siguientes comandos:

#apt-get install subversion
#apt-get install gawk

El siguiente paso consistirá en instalar un SDK, que no es más que un kit de desarrollo que permite crear aplicaciones para un sistema concreto, como pueden ser paquetes de software o plataformas hardware. Para ello en la página de *openwrt* [11] buscaremos la arquitectura que instalamos anteriormente en los *routers*. Una vez encontrada ejecutaremos con permisos de *root* en un terminal:

#svn co svn://svn.openwrt.org/openwrt/trunk kamikaze

A continuación entramos en la carpeta que contiene la copia local, y ejecutamos:

#make menuconfig

En pantalla aparecerá un menú muy similar al que aparecía cuando se iba a crear el nuevo *kernel* de Linux, en el cual el programa de PMIPv6 se pudiese ejecutar. En dicho menú marcaremos con una "M" lo siguiente:

- **Target system:** Broadcom BCM947xx/953xx[2.4]
- **Target profile:** Generic Broadcom WIFI (default)
- **Build SDK**
- **Build the openWRT based toolchain**

Todas las demás opciones no se marcaran, o se desmarcarán en el caso que estén marcadas, ya que el proceso de compilación para crear el SDK consumirá mucho tiempo, y al reducir las opciones a compilar el tiempo consumido será menor. Dichas opciones no serán necesarias, dado que en nuestro caso únicamente queremos crearnos el entorno para poder compilar el programa y que éste se pueda ejecutar en el *router*. En caso de dejarlas, además crearemos la imagen. A continuación se guardan los cambios, y se ejecuta el siguiente comando para compilar:

#make

Una vez que termine este proceso, ya se tendrá generado el SDK, en el directorio: kamikaze/build_dir/target-mipsel_uClibc-0.9.30.1$/OpenWRT_SDK-brcm_2.4-for-Linux-i686-gcc-3.4.6_uClibc-0.9.30.1. Y por tanto, el compilador, con el cual deberemos compilar el programa, se encontrará dentro de ese mismo directorio en: ../staging_dir/toolchain-mipsel_gcc-3.4.6.uClibc-0.9.30.1/usr/bin. A continuación se exportará este *path*, para que cuando se ejecute el comando para compilar el programa, en vez de utilizar el compilador gcc por defecto, que será el que compile para la arquitectura del ordenador en el que se está compilando, se utilice este nuevo compilador.

El último paso que faltaría para obtener el archivo que se pueda ejecutar en el *router*, será compilarlo, para ello se ejecuta el siguiente comando:

#mipsel-openwrt-linux-uClibc-gcc –o attach_detection pmipv6_802.11_attach_detection.c

A continuación se deberá copiar el archivo ejecutable a ambos *routers*. Antes de realizar esta operación será necesaria la configuración tanto de los ordenadores como de los *routers*, ya que se necesitará conectarse desde los ordenadores a los *routers* a través de SSH. Además, será conveniente asegurarse de que dicho archivo tiene permisos de ejecución, para ello, y solamente por si a caso, se ejecuta el siguiente comando para darle todos los permisos a dicho archivo.

#chmod 777 attachdetection

Para poder conectarse por SSH desde cualquiera de los ordenadores que actúan como MAG, es imprescindible asignarle a la interfaz por la que éstos están conectados a los *routers*

una dirección IP perteneciente a la misma subred que la dirección IP que tiene asignada el *router*, recordar que ésta última dirección es la que todos los *routers* tienen por defecto (por tanto a ambos *routers* le corresponderá la misma), 192.168.1.1.

Por ello, y al igual que se hizo con las demás interfaces, en el archivo /etc/network/interfaces, se deberá añadir lo siguiente:

En moscardón:

```
auto eth0_rename
allow-hotplug eth0_rename
iface eth0_rename static
        address 192.168.1.2
```

En saltamontes:

```
auto eth0_rename
allow-hotplug eth0_rename
iface eth0_rename static
        address 192.168.1.3
```

Y, a continuación levantar las interfaces recientemente configuradas, e iniciar de nuevo, todas las conexiones de redes del ordenador. Por ello se ejecuta en ambos ordenadores y por este orden los siguientes comandos:

#ifconfig eth0_rename up
#/etc/init.d/networking restart

Realizadas estas operaciones, los ordenadores ya se podrían conectar mediante SSH al *router*, y desde ahí poder realizar operaciones en el *router*, cómo pueden ser; ejecutar un programa o configurar cualquier parámetro en dicho *router*. Como ya se ha mencionado anteriormente, los dos *routers* tienen la misma dirección IP, con lo cual surge un problema: todos los ordenadores del laboratorio se inician abriendo la misma sesión, por lo que todos tendrán los mismos datos, y al intentar conectarte al primer *router*, se accede sin ningún impedimento, pero al conectarte al segundo *router*, en el archivo /.SSH/known_hosts se ha guardado la IP del *router* y su firma RSA (por seguridad, para evitar suplantación de identidad), y entonces, la firma RSA del primer *router* no coincide con la del segundo, aunque su IP sea la misma, por lo que la conexión falla, al no poder garantizar que el *host* al que tratamos de conectarnos sea él.

Por no modificar la dirección IP de ninguno de los dos *routers*, siempre que se quiera iniciar una conexión SSH desde el ordenador con cualquier *router*, previamente habrá que abrir el archivo dónde se guardan los *hosts* a los que nos hemos conectado anteriormente con su correspondiente firma RSA, y eliminar todo su contenido, para así simular que el *router* al que nos vamos a conectar nunca antes nos hemos conectado, y confirmar que la firma RSA que él nos envía es la correcta. Esto se decidió realizarlo de este modo ya que partimos de la

base que en la red en la que se está trabajando es segura y no existe posibilidad alguna de suplantación de identidad, en caso de no estar seguro de que no pudiese producirse, entonces la mejor solución hubiese sido modificar la dirección IP de alguno de los dos *routers*.

Cuando se ha solucionado este problema, ya se puede conectarse a los *routers* ejecutando, con los permisos de usuario:

#ssh root@192.168.1.1

Nos pedirá contraseña de *root*, ya que accederemos al *router* como ese usuario, y una vez introducida, ya podremos realizar cualquier operación en él, por lo que se podrá proceder a su configuración para que el nuevo programa se ejecute correctamente en ellos. Antes de nada será necesario instalar en ambos *routers* dos paquetes:

- WL: este paquete permite colocar a la placa *wireless* en modo monitor.
- IPV6: es necesario para poder realizar en el *router* operaciones de IPv6.

Procedemos a descargarnos dichos paquetes en uno de los ordenadores. Para ello accedemos a la página de OpenWRT, y los buscamos, dependiendo de nuestra arquitectura y el *firmware* que instalamos. Ellos se encuentran, respectivamente en:

OpenWRT->downloads->kamikaze->8.09.2->brcm-2.4->packages->wl_4.150.10.5.3-3.2_mipsel.ipk

OpenWRT->downloads->kamikaze->8.09.2->brcm-2.4->packages->kmod-ipv6_2.4.35.4-brcm-2.4-1_mipsel.ipk

A continuación se procede a copiar los archivos descargados a los *routers*. Para ello se ejecuta, sin permisos de *root*:

#scp wl_4.150.10.5.3-3.2_mipsel.ipk root@192.168.1.1:
#scp kmod-ipv6_2.4.35.4-brcm-2.4-1_mipsel.ipk root@192.168.1.1:

Al igual que antes, al realizar cada una de las operaciones, será necesaria introducir la contraseña de *root* del *router*. A continuación nos conectaremos, como se ha comentado con anterioridad, al *router* para llevar a cabo la instalación de dichos paquetes en él. Una vez dentro del *router*, se ejecutarán, con este propósito los siguientes comandos:

#opkg install wl_4.150.10.5.3-3.2_mipsel.ipk
#opkg install kmod-ipv6_2.4.35.4-brcm-2.4-1_mipsel.ipk

Con ello, ya estarán instalados. Con lo cual, y para terminar, será necesario copiar el programa que se ha compilado al *router*, y comprobar que su funcionamiento es el esperado.

Para copiarlo se utilizará el mismo comando que ha sido utilizado previamente para copiar los paquetes al *router*:

#scp attachdetection root@192.168.1.1:

Ahora, simplemente se ejecutará el programa (conectados por SSH al *router* desde un ordenador), como un programa normal, de la siguiente manera.

#./attachdetection fe80::227:19ff:feb0:1de 1234

La dirección IPv6 que aparece como parámetro del programa es la dirección IPv6 de la interfaz por la que el *router* se encuentra conectado al MAG. Recordar que en ambos MAG está dirección es la misma ya que fue modificada. El número 1234 es el número de puerto UDP por el que el *router* estará escuchando las nuevas conexiones. Destacar que este puerto deberá ser el mismo en el ordenador que vaya a escuchar los mensajes de nuevas conexiones enviadas por el *router*.

Para ejecutar la otra parte del programa en el ordenador, éste deberá ser compilado como cualquier programa implementado en C. Para ello será compilado, y posteriormente ejecutado de la siguiente forma:

#./attachrcv fe80::227:19ff:feb0:1de eth0_rename 1234

En el cual, la dirección IP es la dirección de acceso local de la interfaz por la cual el mismo está escuchando, "eth0_rename" es el nombre de dicha interfaz, y 1234, es el número del puerto UDP por el que escucha los mensajes de conexión enviados por el *router*, y que, obligatoriamente ha de ser el mismo por el cual el *router* los envía.

Una vez instalado, se comprueba el funcionamiento del mismo, ya que como se verá posteriormente, dicho programa será insertado en la implementación del protocolo PMIPv6, para aumentar su funcionalidad. Cuando ambos programas están siendo ejecutados simultáneamente, en un *router* y en un ordenador, se puede seguir su funcionamiento a través de los mensajes mostrados en pantalla por ambos. Así, por ejemplo, cuando se ejecuta en el *router*, y un nuevo nodo se conecta a él, aparece:

"New MN attachment: MAC address 0:f:b5:e2:46:b4"

En este caso, el nodo conectado corresponde con la interfaz "*wlan0"* de nuestro nodo móvil. Entonces, el *router* le envía el mensaje oportuno al ordenador, y en éste último aparece:

"New event recived: Type 1
MAC address: 0:f:b5:e2:46:b4"

En el caso en el cual dicho nodo se desconecte del *router*, entonces aparecerá en él:

"New MN detachment: MAC address 0:f:b5:e2:46:b4"

Y por lo tanto en el ordenador:

"New event recived: Type 0
MAC address: 0:f:b5:e2:46:b4"

Entonces, se deduce, que el *router* envía un mensaje al ordenador con valor 1 en el campo *type*, cuando detecta que un nuevo nodo se ha conectado a él, y con valor 0 cuando se desconecta de él. Por tanto, además de conocer la dirección MAC del nodo que ha provocado la invocación del envío del mensaje al ordenador, será necesario saber el tipo de evento que ha sucedido, para realizar las acciones pertinentes.

F. Guía de usuario

El programa desarrollado no requiere gran interacción por parte del usuario, dado que se trata de un programa que estará siendo ejecutado continuamente en el ordenador, sin que probablemente el usuario sea consciente de ello.

Por eso mismo, lo único necesario a realizar por el usuario será la creación del fichero de configuración (dado que sin él no se obtendrá la información necesaria para la ejecución) y poner dicho programa en funcionamiento.

Creación del fichero de configuración

Como se ha comentado en el capítulo 4, este fichero debe ser incluido en el mismo directorio en el que se encuentra el archivo ejecutable del programa, y ha de ser llamado con el nombre "pmip.conf". Si cualquiera de estas dos restricciones no se cumplen el programa no podrá ser ejecutado con éxito, dado que no se podrá leer el fichero y no recopilará la información inicial necesaria para ello.

El archivo de configuración ha de ser un archivo de texto plano, en el cual, en cada línea se incluya el nombre de la variable, un espacio y a continuación el valor que tomará esa variable. No se debe separar las líneas mediante ";" , "." o cualquier otro símbolo. En ese mismo sentido, entre el nombre de la variable y su valor únicamente debe existir un espacio en blanco.

Los nombres de las variables que pueden ser incluidos, y su significado, han sido explicados previamente en el apartado 4.5. A modo recordatorio se enunciarán de nuevo:

{coa, coa_ip, interface_coa-mn_num, interface_coa-mn_wor, interface_coa-lma_wor}[1]

Estas cinco variables harán mención a la información relativa al MAG, por lo tanto será necesario que aparezcan una y solamente una vez.

Las variables relacionadas con la información de los diferentes MN, podrán aparecer de 0 a N veces, dependiendo el número de MN a los que se les pueda gestionar la movilidad. Éstas son tres, y deberán aparecer cada una de ellas el mismo número de veces. Es decir, en el caso que se quiera añadir un nuevo MN se deberá añadir toda su información. En el caso de no incluir alguna de las variables, el nodo no será dado de alta como posible nodo al que proporcionarle PMIPv6. Además, deberán encontrarse las tres variables seguidas una de otra, no se podrá mezclar información de varios MN. Es decir, no se podrá incluir, por ejemplo,

primeramente todas las direcciones HW (utilizado como identificador) de todos los MN que se quieran incluir, y a continuación los demás datos. Estas variables son:

{mn_ip, mn_hw, mn_lma}$^{0\text{-}N}$

El orden de dichas variables es indiferente dentro de cada grupo, siempre y cuando se cumplan las restricciones anteriormente enunciadas.

Un ejemplo de un archivo de configuración válido, correspondiente con el entorno de pruebas que se ha utilizado, en el que se incluyen dos MN, sería:

```
coa 2001::3
coa_ip fe80::227:19ff:feb0:1de
interface_coa-mn_num 3
interface_coa-mn_wor eth0_rename
interface_coa-lma_wor eth1

mn_ip fe80::20f:b5ff:fee2:46b4
mn_hw 00:0f:b5:e2:46:b4
mn_lma 2001::1
mn_mac fe80::21c:bfff:fe37:4c37
mn_hw 00:1c:bf:37:4c:37
mn_lma 2001::1
```

Por último, cabe remarcar que un fichero de este tipo debe ser incluido en cada nodo que actúe con el rol de MAG del entorno en el que el programa vaya a ser utilizado.

Ejecución del programa

La otra acción que el usuario deberá realizar es poner en funcionamiento dicho programa. Para ello, dado que se ha creado un archivo *makefile*, en caso de que el archivo ejecutable no exista, deberá crearlo compilando el programa mediante el comando:

#make

Una vez que se creado el archivo ejecutable del programa, únicamente habrá que ejecutarlo. El ejecutable creado, debido a las especificaciones del *makefile,* es nombrado con el nombre "*pmipv6"*. Por lo tanto, se ejecutará el comando:

#./pmipv6

Como se puede observar, el programa no necesita ningún argumento para que funcione correctamente, dado que toda la información necesaria le ha sido facilitada mediante el fichero de configuración.

Se puede realizar otra acción gracias al archivo *makefile*, aunque es menos común que el usuario la realice, que será la de eliminar el archivo ejecutable y todos los ficheros "*.o", es decir, archivos objeto, del programa. De este modo, únicamente se tendrán los archivos del código fuente del programa, es decir los "*.c" y "*.h". El comando a realizar en caso de querer realizar esta acción es:

#make clean

Resulta importante destacar que todos estos comandos han de ser ejecutados dentro del directorio donde se encuentren todos los archivos con el código fuente perteneciente al programa.

Además, para que el programa se ejecute correctamente, ha de ser ejecutado con permisos de "superusuario", es decir con el usuario *root*, ya que éste consta de un mayor número de privilegios. Esta condición es necesaria debido a la utilización de *sockets RAW*, ya que trabajan estrechamente con el *kernel* de la máquina, y este usuario es el único que puede crear este tipo de conectores. Por lo general, para acceder a este tipo de usuario, deberá ser necesario ejecutar el comando:

#sudo –s

El *flag* "s" indicará que todas las acciones que se ejecuten a continuación serán realizadas como usuario *root*. El resultado de este comando sería el mismo que si se escribiese "sudo" delante de cada comando, con lo que se indicará que esa acción se ejecutará con los mismos permisos que *root.*

Una vez que se ejecute ese comando, el sistema operativo pedirá contraseña. Será necesaria introducirla para confirmar que se dispone de esos privilegios. Una vez autenticado, se podrá proceder a la ejecución del programa como se ha descrito en los párrafos anteriores.

G. Trabajo otra implementación

I. Instalación

El programa utilizado para la realización de este proyecto, fue desarrollado por la Universidad de Helsinki. La versión utilizada es la 2.0.2 y data de Febrero del 2010. Para un correcto funcionamiento, se ha procedido a realizar las operaciones descritas a continuación. Matizar que dichas operaciones únicamente serán aplicadas a los MAGs y al LMA. En el nodo móvil no será necesario debido a que no se debe instalar ningún software relacionado con movilidad IP en él. Todos los equipos utilizados tienen instalado como sistema operativo Ubuntu 9.10 de 32 bits, con el *kernel* 2.6.31.

Instalación programa PMIPv6

Una vez que el kernel haya sido modificado y compilado para ser adaptado al protocolo PMIPv6, y se haya arrancado el equipo con el nuevo *kernel*, ya se puede instalar el programa de PMIPv6. Para ello, el primer paso será copiar la carpeta dónde se encuentra todo el código fuente al directorio "usr" del ordenador. A continuación, y dentro del directorio dónde se encuentra el programa se ejecutará un script de configuración:

#CPPFLAGS='-isystem usr/src/linux' ./configure –enable-vt

Debido a que las fuentes de nuestro *kernel* no se encuentran en el directorio por defecto usr/include/Linux es necesario a la hora de la configuración indicar dónde se encuentran. En nuestro caso dicho directorio es el indicado (usr/src/linux). Si se encontrasen en el directorio por defecto únicamente habría que ejecutar el comando "./configure –enable-vt".

Dado que no se desactiva en este paso los mensajes para depurar el código, es decir los que van informando de que está realizando el código, cuando se ejecute el programa dichos mensajes aparecerán. Para desactivarlos habría que indicarlo mediante la opción "—disable-debug".

La opción "—enable-vt", como su propio nombre indica permite activar la terminal virtual, para facilitar la resolución de problemas del programa. Es posible acceder mediante *telnet* al puerto 7777 a varios estados en los que se encuentre el programa en ejecución. Es posible cambiar este puerto mediante la opción "—vt-service= nº de puerto".

Si se completa con éxito esta operación, se deberá generar un archivo *makefile* que al ejecutarlo se compile el programa. El *makefile* que se genera es erróneo, por lo que se

sustituirá dicho archivo por otro que nos ha sido facilitado. Por lo tanto, se copiará el nuevo *makefile* a la carpeta "src" del directorio en el cual se encuentra el programa.

A continuación se compila el programa.

#make

Tras la ejecución de este comando aparecían varios errores, por lo que el programa no se compilaba. Para solucionarlo fue necesaria la instalación de algunos paquetes más de Linux. Su instalación fue del siguiente modo:

#apt-get install automake

Este paquete contiene programas para la generación automáticas de *makefiles* que se generan con *autoconf* [10].

#apt-get install autoconf

Este paquete contiene los programas necesarios para la generación automática de guiones utilizados por el intérprete de comandos para la configuración del código fuente [10].

#apt-get install perl

El contenido de este paquete es el Lenguaje Práctico de Extracción e Informe [10].

#apt-get install m4

El paquete m4 contiene un procesador para macros [10]. Es decir, expande y ejecuta las macros que se encuentre en el fichero indicado.

#apt-get install indent

Este paquete permite formatear los códigos fuentes escritos en C según varios estilos [10].

#apt-get install flex

El contenido del paquete flex son las herramientas necesarias para generar programas que reconozcan patrones de texto [10].

Una vez instalados todos ellos, el proceso de compilación del programa se completa correctamente. Por lo tanto, se procede a su instalación, ejecutando:

#make install

Este proceso termina con éxito, por lo que tanto en el directorio usr/local/sbin como en el mismo directorio dónde se encuentra el código fuente se encuentra un archivo ejecutable llamado *mip6d.* Para la ejecución de dicho programa, bastará introducir en la *Shell* el siguiente comando:

#./mip6d

Por último, y para que el programa funcione correctamente, habrá que configurar todos los equipos involucrados en las pruebas del programa para que actúen como un *router* IPv6, es decir, que puedan recibir paquetes que no estén destinados a ellos, y sean capaces de encaminarlos correctamente hacia el destino. Para ello se abre con permisos de escritura el archivo "etc/sysctl/.conf", y en él nos encontramos con lo siguiente:

#uncoment the next line to enable packet forwarding for IPv6
#net.ipv6.conf.all.forwarding=1

Tal y como dice el archivo se descomenta la segunda línea. A continuación se reinicia el equipo para que dichos cambios surjan efecto. Para comprobarlo se puede acceder a la variable "*forwarding*" en proc/sys/net/ipv6/conf y mirar su valor, que efectivamente es 1.

II. Configuración

Una vez preparado el entorno de pruebas como se ha descrito en el apartado 5.1, se debe configurar cada elemento de dicho entorno para el correcto funcionamiento del programa que simula el funcionamiento del protocolo PMIPv6. Para ello será necesaria la creación de dos ficheros de texto (dependiendo del tipo de nodo en el que se encuentren) denominados *mip6d.conf* y *MAC.conf*, ambos ubicados en un directorio por defecto, como específica el programa, que en este caso coincide con: */usr/local/etc/*.

Así, el fichero *mip6d.conf* contendrá la información necesaria para que cada nodo desempeñe su función correspondiente dentro del protocolo PMIPv6, y en el archivo *MAC.conf* se encontrará la información de los posibles nodos móviles que se pueden conectar al sistema.

A continuación se describe con mayor nivel de detalle el contenido de cada fichero en el tipo de nodo en el que se encuentren. Además de una explicación de porqué se han seleccionado dichas variables, y de su valor.

LMA

El ordenador en el que se ejecutará la funcionalidad del LMA dentro del protocolo será "Mariposa". Dado que la aplicación utilizada para emular el funcionamiento de PMIPv6 es una adaptación de otra aplicación utilizada para simular el comportamiento del protocolo

predecesor, recordemos MIPv6, la funcionalidad del LMA de PMIPv6 se desarrollará bajo el rol de HA correspondiente al protocolo MIPv6.

Para el correcto funcionamiento de este nodo, únicamente será necesario el fichero *mip6d.conf*, dado que la información correspondiente a los nodos móviles se le será facilitada mediante el MAG a través de un PBU.

Además del rol que desempeña el nodo en el sistema, para el caso del LMA será necesario especificar la interfaz por la cual este nodo está conectado a ambos MAGs. Será precisa también la activación de dos *flags* para permitir tanto la llegada de solicitudes de prefijos ("*Prefix Solicitations*") así como, el anuncio del prefijo asignado ("*Prefix Adverstisement*").

Asimismo, se le especificará la dirección MAC de la interfaz del MAG por la cual un nodo se conecta a éste último. Recordemos que ambos MAG tienen la misma dirección MAC en la interfaz por la que se permite que un nodo móvil se conecte a él. Esto es debido a que en uno de los MAG se procedió a un cambio de dicha dirección.

Por lo tanto, en nuestro caso el fichero de configuración *mip6d.conf* en el ordenador que actúa como LMA será:

```
NodeConfig HA;

DebugLevel 10;

Interface "eth2";
LMAInterfaceMAG "eth2";

SendMobPfxSols enabled;
SendUnsolMobPfxAdvs enabled;

FixedMAGLinkLocalAddressOnAllAccessLinks fe80::227:19ff:feb0:1de;
```

Mencionar que la opción "*DebugLevel*" simplemente servirá para que el programa muestre por pantalla los diferentes pasos que va realizando, por lo tanto se podrá obtener una traza de las diferentes funciones que el programa va ejecutando, así como el valor de ciertas variables en un determinado momento. Para conseguirlo, esta opción deberá tomar un valor entero mayor de 0. El programa mostrará siempre lo mismo independientemente de dicho valor.

MAGs

Como ya se ha comentado anteriormente, el número de ordenadores en los que se ejecutará la funcionalidad MAG será de dos, y se llevará a cabo en los ordenadores "Saltamontes" y "Moscardón". Para un funcionamiento adecuado de ambos, se necesitarán los

dos ficheros de configuración mencionados anteriormente. En este caso, la funcionalidad del nodo MAG será llevada a cabo bajo el rol de Nodo Móvil del protocolo MIPv6.

Al igual que en el caso del LMA, será necesario especificar la dirección MAC correspondiente a la interfaz por la cual se podrán conectar a ellos cualquier nodo móvil autorizado. De la misma manera, será necesario indicar el nombre de dicha interfaz.

Además, en el caso de los nodos MAG, será necesario indicar la dirección IPv6 de acceso global de la interfaz por la que están conectados al LMA, es decir la CoA, que será un extremo del túnel, que se creará para establecer la comunicación entre el LMA y el MAG. Por lo tanto, sin dicha dirección, esa comunicación no sería posible.

Para que un nodo móvil esté autorizado a utilizar las funcionalidades que el protocolo PMIPv6 le proporcione, el MAG al que se conecta debe conocer determinados datos de él. El conjunto de dichos datos se denominará perfil del nodo móvil. Entre ellos, el más importante será el identificador, en este caso corresponde con la dirección MAC de la interfaz correspondiente al nodo móvil por la cual éste se conecta al MAG. Además será imprescindible, indicar la dirección IPv6 de acceso global del LMA que gestionará la movilidad de dicho nodo móvil, así como el prefijo que se le quiera asignar a dicho nodo móvil. Asimismo, será necesario activar la funcionalidad de PMIPv6, dado que si no, el MAG no le proporcionará sus servicios a dicho nodo móvil.

Dado que no es necesaria una optimización de las rutas por las que se envíen los paquetes desde el MAG al nodo móvil, ni a cualquier otro nodo que intente comunicarse con él, se desactivará esa opción, dado que por defecto está activada.

Tampoco será preciso, añadir el *bit* de asentimiento en los mensajes enviados desde el nodo móvil al nodo que trata de comunicarse con él (CN), por lo que esta opción será también desactivada.

Se ha decidido que el nodo móvil descarte los mensajes de control procedentes del LMA (HA en MIPv6), debido a que estos mensajes normalmente no están protegidos mediante IPsec, y se podría suplantar la identidad del LMA fácilmente. Además, si el nodo móvil acepta algún mensaje corrupto, fácilmente dejaría de funcionar, y por lo tanto, desencadenaría en un ataque de denegación de servicio (DoS). Esto no afectaría a nuestro entorno de pruebas, ya que es bastante pequeño y todos los nodos están siendo controlados en cada momento, pero dichas condiciones no se darían en un entorno real, y llevar a cabo el ataque sería asequible.

Cuando el nodo móvil cambie de ubicación, es decir cambie de punto de acceso a la red, deberá enviar un único mensaje a su antiguo punto de acceso indicando que él ya no es alcanzable por esa ruta. Esto deberá suceder siempre después de que el nodo móvil haya recibido un mensaje RA del nuevo punto de acceso. En caso de que la opción *"MnRouterProbes"* esté establecida a 0, el nodo móvil se moverá directamente al nuevo punto de acceso.

Al resultar seguro nuestro entorno de pruebas no será necesario proteger todos los mensajes intercambiados entre el nodo móvil y el LMA mediante IPsec, por lo que esa opción será desactivada, ya que por defecto está activada.

El fichero *mip6d.conf* será el mismo en ambos MAG, a excepción de la opción "*MAGEgressGlobalAddress"* que cada uno le corresponderá una. El ejemplo de fichero mostrado a continuación corresponde con el fichero perteneciente al MAG-1, "Saltamontes". En el fichero perteneciente al MAG-2, "Moscardón", este campo tomará el valor de "2001::2".

Al llamarse igual todas las interfaces de ambos MAGs el campo *interface* será el mismo en los dos ficheros de configuración. Tampoco cambian los nodos móviles que se pueden conectar a los MAGs.

Por lo tanto el fichero de configuración de los MAGs será como sigue:

```
NodeConfig MN;

DebugLevel 10;

MAGEgressGlobalAddress 2001::3;

Interface "eth0_rename";

DoRouteOptimizationCN disabled;

DoRouteOptimizationMN disabled;

UseCnBuAck disabled;


MNIdentifier "00:0f:b5:e2:46:b4" {
    PMIPEnabled 1;
    SupportedConfigurationMode 0;
    LMAAddress 2001::1;
    HomeNetworkPrefix1 2002::/64;

}

MNIdentifier "00:1c:bf:37:4c:37" {
    PMIPEnabled 1;
    SupportedConfigurationMode 0;
    LMAAddress 2001::1;
    HomeNetworkPrefix1 2004::/64;
}

MnDiscardHaParamProb enabled;

Mn*Router*Probes 1;
```

```
UseMnHaIPsec disabled;

FixedMAGLinkLocalAddressOnAllAccessLinks fe80::227:19ff:feb0:1de;
```

Matizar, que en este caso, el fichero de configuración contiene los datos de dos nodos móviles, ya que, aunque sea un único MN en el entorno de pruebas dispone de dos tarjetas de red inalámbricas, y será necesario disponer de los datos de cada una, para poder proporcionarles a ambas los servicios de PMIPv6.

A parte del fichero *mip6.conf* en los nodos MAG, será necesario el fichero *MAC.conf*, ubicado en el mismo directorio por defecto. Este fichero aporta más información útil acerca de los nodos móviles que se pueden conectar al MAG que contiene este archivo.

Este fichero contiene cuatro campos por cada nodo móvil:

- MAC Address: Cómo su propio nombre indica, corresponde con la dirección MAC del nodo móvil. Ésta ha de ser la misma que se especificó en el fichero *mip6d.conf*.
- MN ID: Este campo será un identificador numérico asignado al nodo móvil. Tendrá el mismo formato que una dirección MAC.
- MUHO: Si a un nodo móvil se le puede asignar más de un prefijo de su *home network*, entonces este campo tomará el valor T (True). En caso contrario su valor será F (False). Por lo tanto indicará la posibilidad de que un nodo pueda tener más de un prefijo asignado para la comunicación con otros nodos.
- PREFIJO: Al igual que en el fichero *mip6d.conf*, este campo indicará el prefijo que se le asignará al nodo móvil en el instante que entre a formar parte de la red.

Por lo tanto, el fichero *MAC.conf* en ambos MAG será idéntico, y corresponde con:

```
## MAC ADDRESS     | MN ID              | MUHO | PREFIX |
------------------|--------------------|------|--------|
00:0f:b5:e2:46:b4 | 11:11:11:11:11:11 | T    | 2002   |
```

Figura 35: Ejemplo fichero configuración MAC.conf

MN

Como se ha mencionado anteriormente, la novedad más significante de PMIPv6 con respecto a MIPv6 es que el nodo móvil no tiene que llevar a cabo ninguna acción, en cuanto a movilidad se refiere, ya que el no es consciente en ningún momento que está cambiando de posición. Por lo tanto, y al igual que no es necesario ejecutar ningún tipo de SW para el correcto funcionamiento de PMIPv6, no será necesario ningún tipo de configuración, además de ciertas opciones relativas al ordenador, y mencionadas en el Anexo C.

Resultados:

Una vez se haya realizado la preparación de todos los equipos implicados en el entorno de pruebas, y se hayan creado los ficheros descritos anteriormente en este mismo apartado, ya se puede ejecutar el programa, y obtener los primeros resultados.

Siempre que el programa de PMIPv6 sea ejecutado, y dado que la opción "*DebugLevel*" en todos los ficheros es mayor de 0, en la pantalla en la que se ejecute la aplicación aparecerá cierta información sobre las acciones que éste está realizando, como puede ser las diferentes funciones que invoca así como el valor de ciertas variables importantes en un determinado momento.

Al inicio del programa, siempre aparecerán los valores asignados, mediante los ficheros de configuración, a las variables necesarias inicialmente para su ejecución. En el caso en que no se le haya asignado ningún valor, éstas tomarán el valor por defecto que los desarrolladores anteriores consideraron oportunos.

III. Funcionamiento

Una vez realizado todos los pasos anteriores, el programa ya debería funcionar correctamente. Para comprobar que el programa realmente lleva a cabo lo que se desea, se ejecutará el comando *ping6* (igual que *ping*, pero para ser utilizado en IPv6) en el nodo móvil con dirección destino el LMA. Este comando sirve para comprobar que ambos equipos están conectados y son accesibles dentro de la red creada. Los paquetes enviados están definidos en el protocolo ICMP, y contendrán únicamente un *echo request* y *echo reply*. En este caso, es el nodo móvil el que enviará la solicitud, y el LMA el que contestará.

Además será necesario comprobar los mensajes intercambiados entre los nodos MAG y el nodo LMA. Para ello, se usará el programa *Wireshark*. Este programa es uno de los denominados *sniffers*, que "capturará" todo el tráfico de paquetes que son enviados y recibidos por la interfaz que se desee, y lo mostrará de un modo legible para el usuario con el fin de que posteriormente puedan ser interpretados. Este intercambio de mensajes deberá coincidir con los mensajes enviados entre el LMA y el MAG mostrados en las [*fig.* 2] y [*fig.* 3].

A continuación se muestra un diagrama de interacción entre los elementos del entorno de pruebas. En él se representará el elemento que realiza una acción, y como interacciona con los demás mostrando los mensajes enviados.

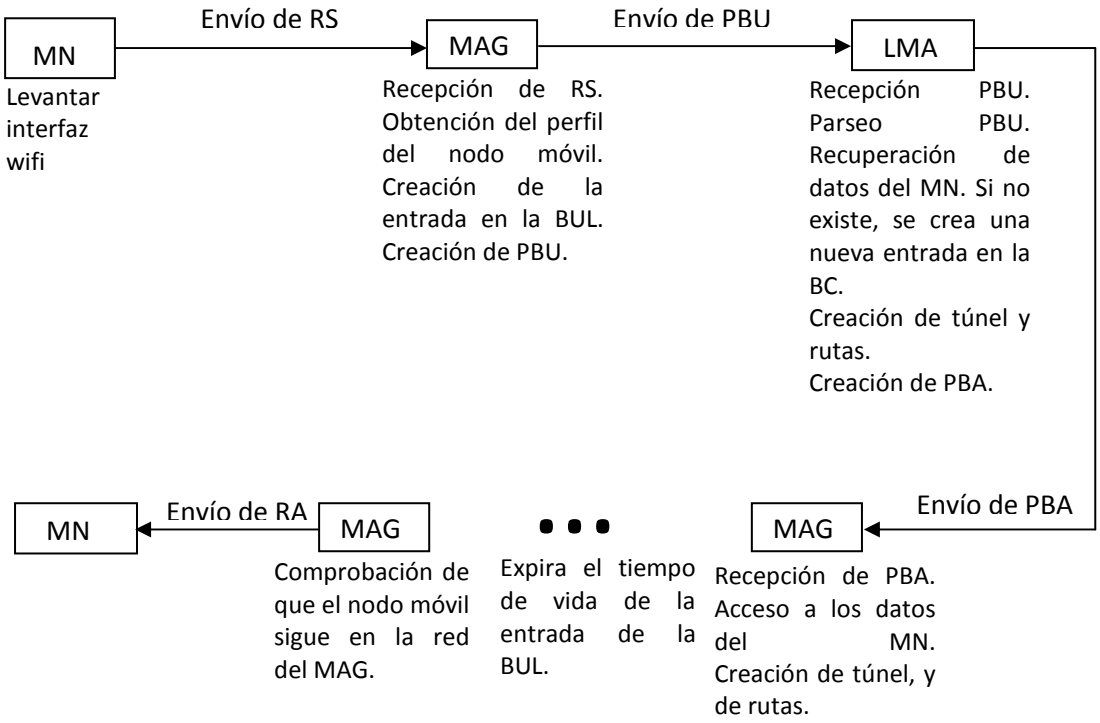


Figura 36: Funcionamiento implementación PMIPv6

El nodo móvil continúa en la red del MAG. Entonces:

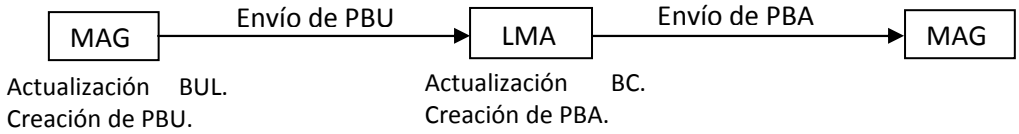


Figura 37: Funcionamiento implementación PMIPv6 cuando MN continúa en la red

Una vez que el MAG recibe el PBA enviado por el LMA, éste volvería a esperar a que la entrada en la BUL caducara, a continuación se comprobaría de nuevo que el nodo móvil sigue en su misma red, y se repetiría el mismo proceso una y otra vez. En el caso de la implementación de PMIPv6 utilizada, para la comprobación de que el nodo móvil continúa conectado a su red ejecuta el protocolo DAD. Cómo se ha explicado anteriormente, este protocolo es utilizado para comprobar que en una misma red no existan dos nodos con la misma IPv6. Para ello se enviará un mensaje NS a la dirección escogida, y si se recibe respuesta NA, entonces es que esa dirección está siendo utilizada. Lo mismo sucede en nuestro caso. El MAG enviará un mensaje a la dirección del nodo móvil, si éste le responde será porque aún está conectado a su red, en caso contrario, el nodo móvil habrá cambiado su ubicación.

Una vez que se procede a probar el funcionamiento del programa, se observa que aunque la comunicación entre el MN y el LMA es correcta, solamente está activa durante un período de tiempo muy reducido. Además, en la traza de ejecución del programa el MAG le envía continuamente PBUs al LMA, y éste último le contesta con PBAs. Examinando con detalle

la traza que aparece en la terminal de ejecución del programa, llama la atención que el tiempo de validez del prefijo siempre sea -1, si se mira el código, se puede ver que es el valor con el cual se ha inicializado dicha variable. Por lo tanto, en un primer momento se piensa que el fichero de configuración creado anteriormente está incompleto.

Analizando la estructura y las posibles variables de entrada que están permitidas en el fichero "mip6d.conf", especificado en el fichero "gram.y", se encuentra una posible variable de entrada denominada *HOMEPREFIXLIFETIME*, lo cual da a entender que servirá para especificar desde el principio el tiempo de vida del prefijo asignado al nodo, o lo que es igual, el tiempo de validez de la entrada de dicho nodo en la BUL perteneciente al MAG. Por lo tanto, se incluye en el fichero de configuración. Pero el resultado no es el esperado, ya que se produce un error en el análisis léxico cuando se lee el fichero de configuración. Además de no leer esta variable, las demás variables declaradas después de ella no se configuran con el valor que se desea, si no con el valor por defecto. Con lo cual, no se consigue el efecto deseado.

Destacar, que se intentó de todas las maneras que se creen posibles que dicha variable fuese incluida en el fichero de configuración, pero siempre aparecía el error del análisis léxico en la línea en la que la variable era declarada. Por lo tanto, se tomó una decisión más drástica, que fue la de modificar el código fuente para lograr que el valor de la entrada del nodo móvil en la BUL fuese algo más duradera, y conseguir de este modo un funcionamiento más estable.

Por lo tanto, en el archivo que se realizan las funciones del MAG, "mn.c", en la función que se ejecuta cuando se recibe un *RS*, se observa que también se establece el tiempo de vida del prefijo, puesto que es ahí donde se crea la entrada de la BUL. En la línea 409, y siguientes se observa:

```
if (tsisset(mnProfil->homeNetworkPrefixLifetime)){
      struct timespec maxBindingLft;
      tsset(maxBindingLft, MAX_BINDING_LIFETIME, 0);
      lifetime      =      tsafter(mnProfil->homeNetworkPrefixLifetime,
maxBindingLft) ? mnProfil->homeNetworkPrefixLifetime : maxBindingLft;
      }
else
      tsset(lifetime, 420, 0);
```

En este fragmento de código, se puede apreciar que el tiempo de vida del prefijo se establece eligiendo el mínimo valor entre el tiempo de vida que le ha sido asignado al principio a su perfil, y el tiempo máximo que puede perdurar una entrada en la BUL. Este tiempo máximo se establece mediante la constante *MAX_BINDING_LIFETIME*, y está prefijada a 420 segundos. En caso de que no se le haya asignado ningún valor a esta variable en el perfil del nodo móvil, entonces se establecerá al valor máximo permitido, es decir 420.

En un principio parece que este fragmento debería realizar correctamente el establecimiento del tiempo de vida de la entrada en la BUL, pero no es así. Dado que no es viable introducir en el fichero de configuración un valor para esta variable, ésta tomará su

valor por defecto, que es -1, y este valor es siempre menor que 420, por lo tanto el valor del tiempo de vida de la entrada de la BUL en la MAG, siempre será -1, y por tanto, estará continuamente expirando, por lo que enviará continuamente PBUs al LMA indicando la actualización de la entrada.

Para solucionarlo, finalmente se decide por dejar única y exclusivamente la última línea del fragmento de código mostrado anteriormente, por lo tanto esta variable siempre se establecerá a 420 segundos. Puesto que al hacer pruebas se desperdicia mucho tiempo esperando 420 segundos para que se produzca la renovación de la BUL, se ha dejado fijada en 60 segundos.

Una vez arreglado este pequeño inconveniente, nos aseguramos que efectivamente el programa actúa como debe actuar. Para ello, levantamos una interfaz WIFI del MN, y lo conectamos a la red WIFI de un *router* conectado a su vez a un MAG. Comprobamos que efectivamente el MN se puede comunicar con el LMA (como se ha mencionado al inicio de este apartado esa comprobación se realiza con el comando *ping6*). Pasados 57 segundos, y a 3 de que la entrada caduque, el MAG comprueba mediante DAD que el MN está aún conectado a su red, por lo que renovará su entrada en la BUL y enviará un PBU al LMA, para que éste también sea actualizado. Esta operación se realiza a los 57 segundos y no a los 60, porque la entrada se eliminará 60 segundos después de ser creada si no se actualiza, por lo tanto, los trámites para que la entrada de la BUL sea actualizada deben realizarse antes de que sea borrada.

Además, se deberá comprobar que la entrada no será renovada si el MN abandona la red por la cual está conectado al MAG. Por lo tanto, se actuará de la misma manera que anteriormente, se levantará la interfaz WIFI del MN, y se conectará al *router* previamente conectado al MAG, se comprueba que el MN y el LMA pueden comunicarse, y a continuación se desconecta la interfaz WIFI del MN. El comportamiento del programa en este caso, es tal y como especifica el protocolo PMIPv6. El MAG sigue teniendo en la BUL la entrada del MN, pasados 57 segundos de que dicha entrada fuese creada, el MAG procede a realizar DAD, obteniendo un resultado positivo, pues el nodo ya no se encuentra en la red del MAG, y por tanto dicha dirección podría ser utilizada por otro nodo en esa red. Detectada la ausencia del MN en la red, el MAG procede a borrar la entrada de la BUL y a eliminar el túnel que le unía al LMA que gestionaba la movilidad del MN, puesto que no existe otro MN que utilice dicho túnel. Además enviará un mensaje PBU al LMA para que éste también elimine el otro extremo del mismo túnel. En ambos nodos, MAG y LMA, las rutas para acceder al MN también son eliminadas.

El siguiente paso será comprobar que efectivamente, la movilidad del MN entre los dos MAGs del entorno de pruebas se ejecuta satisfactoriamente. Para ello, los pasos a seguir serán muy parecidos a los ejecutados anteriormente. Primeramente se levantará la interfaz WIFI que estamos utilizando del MN, y se conectará a uno de los dos *routers* conectados a su respectivo MAG. Se comprobará que el MN se puede comunicar con el LMA, y a continuación se desconectará la interfaz WIFI del MN. Posteriormente se vuelve a levantar y se conecta al otro

router conectado al otro MAG, y se comprobará que efectivamente la comunicación con el LMA se sigue produciendo.

Para comprobar esto, es necesario levantar y desconectar la interfaz WIFI del nodo móvil para que el MAG se percate de que un nuevo MN se ha conectado a su red, y así iniciar los trámites para hacer posible la comunicación con el LMA. El MN en cuestión, debe mandar un RS, siendo ésta la única manera de conseguirlo. Ya que si simplemente conectamos el MN a un *router*, y lo desconectamos de él, y lo conectamos al otro *router*, el mensaje enviado no será un RS.

El procedimiento anteriormente explicado para comprobar la movilidad se llevará a cabo en dos situaciones. Una, cuando ya haya caducado la entrada del MN de la BUL perteneciente al MAG al que estaba conectado, y por tanto se haya borrado el túnel y las rutas pertenecientes a dicho MN, y otra cambiando de MAG cuando aún sigue activa la entrada de la BUL del primer MAG al que nos conectamos.

Cuando forzamos la primera situación, el programa reacciona como se esperaba. La comunicación inicial entre el MN y el LMA es correcta, y cuando expira la entrada de la BUL del primer nodo al que nos hemos conectado, se elimina tanto las rutas como el túnel, de ambos nodos, MAG y LMA. A continuación al levantar de nuevo la interfaz del MN y conectarlo al otro *router*, se ejecutan los mismos pasos anteriormente mencionados, y por lo tanto, el MN y el LMA pueden volver a comunicarse.

El problema surge cuando nos cambiamos de punto de acceso para comunicarnos con el mismo LMA. Inicialmente, y según la traza que la terminal muestra en el MAG dónde se está ejecutando el programa, parece que las acciones que lleva a cabo son las mismas que se ejecutaban en el caso anterior, y en efecto lo son, pero la comunicación entre el MN y el LMA no es posible. Además, también se observa que llegados a un punto, el MAG rechaza la conexión del nodo móvil, y procede a eliminar todo lo relativo a él.

Examinando con mayor detalle la misma traza, llama la atención que el campo estado del PBA que envía el LMA al MAG es “157”, cuando en el código aparece que para que todo funcione correctamente es necesario que dicho campo sea “0” ó “1”. Esto, es simplemente una notación a la hora de la implementación del protocolo, puesto que inicialmente este campo está establecido a 0, y al PBU recibido se le van realizando determinadas comprobaciones para verificar que se trata de un PBU correcto, y no de uno corrupto. Entonces, cuando se detecta un error en el PBU, se van estableciendo distintos valores para dicho campo. Los valores que determinan si el PBA generado por el LMA es correcto serán los ya mencionados.

Por lo tanto, se ve en el código fuente de la aplicación que dicho valor del campo estado del PBA es un error del sello temporal, ya que éste no era válido. En un primer momento se estudió la forma de crearlo, pero se observó que era correcto, ya que se utilizaba la hora del ordenador en el que se estaba ejecutando el programa. La única opción entonces,

era que el sello temporal del PBU enviado por el segundo MAG al que se conectó el MN, fuese anterior al sello temporal del PBU que generó el primer MAG al que se conectó nuestro MN. Y efectivamente, así era, puesto que el reloj del segundo MAG marcaba una hora anterior al reloj del primer MAG.

Entonces, se dedujo que todos los ordenadores en el entorno de pruebas debían estar sincronizados para evitar otro posible problema con los sellos temporales. En un primer momento, ésta operación se realizó manualmente. Es decir, cambiar la hora de los relojes, y establecer en todos la misma.

Realizada esta última operación, se puede comprobar que el cambio de ubicación del MN se lleva a cabo correctamente. Si dejamos funcionando el comando *ping6* en el MN, se observa como la comunicación entre el LMA y él mismo es interrumpida mientras se lleva a cabo las operaciones necesarias para conectarse a otro MAG, pero a continuación vuelve a establecerse. Esto se realiza en un período inferior a los 60 segundos necesarios para que expire la entrada de la BUL en el primer MAG al que nos conectamos. Esta prueba se puede realizar tantas veces como se desee y siempre la comunicación vuelve a producirse entre ambos extremos.

Si nos fijamos en la traza que se muestra en la terminal del nodo en el que se está llevando a cabo la funcionalidad de LMA, se puede observar que el túnel creado entre éste LMA y el primer MAG al que se conectó el MN, no es eliminado, sino que simplemente se modifica el final del mismo, es decir, el nuevo final coincidirá con el MAG al que el MN se conectó por segunda vez, lo cual indica un movimiento del MN. Esto se considera correcto, y además, óptimo, ya que al no eliminarlo y volverlo a crear se ahorra tanto tiempo como recursos del ordenador.

IV. Aumento de funcionalidad

Según lo comentado en el Anexo E, el siguiente paso a desarrollar en el proyecto será ampliar la funcionalidad de la implementación del protocolo. Esta ampliación consistirá en hacer que dicha implementación sea capaz de llevar a cabo la funcionalidad de PMIPv6, en el momento que se detecte un nuevo nodo en la red de cualquiera de los MAG. Como se ha explicado con anterioridad, la implementación de la que se parte inicialmente únicamente responde a mensajes del tipo RS. Estos mensajes son enviados por el nodo móvil única y exclusivamente cuando se levanta una interfaz.

Se considera que esta nueva funcionalidad es necesaria dado que de este modo el protocolo se podrá ejecutar en un entorno que se asemeje más a la realidad. Un usuario que desee conectarse a una red, la primera vez que lo vaya a realizar sí que levantará la interfaz de su equipo, pero en sucesivas ocasiones, solamente se ocupará de conectarse, enviando de esta forma otro mensaje en lugar de un RS. No se preocupará de desconectar, y posteriormente

conectar la interfaz de nuevo cada vez que se cambie de punto de acceso. Además, en muchos casos el usuario de este tipo de servicios ni si quiera será consciente de que ha cambiado de punto de acceso a la red, siendo éste el objetivo principal que se desea obtener con la movilidad IP.

Recordar que, PMIPv6 se crea para mejorar MIPv6, y unos de sus principios básicos es que el MN no sea consciente del cambio de punto de acceso a la red, y debido a ello, de su cambio de dirección IP.

Con lo cual, el nuevo objetivo a conseguir, será que el programa se comporte de igual manera que cuando se recibe un RS, pero recibiendo otro tipo de mensajes que indiquen que un nodo se ha conectado a la red del MAG. Con el mismo comportamiento nos referimos al descrito en el apartado anterior de este mismo anexo. Para ello, en cuanto se reciba un mensaje de conexión, resumiendo, se deberán llevar a cabo las siguientes acciones:

- Comprobar si el nodo móvil está autorizado a usar PMIPv6.
- En caso afirmativo, el MAG le asignará el prefijo correspondiente, actualizará su BUL y enviará un PBU al LMA oportuno.
- El LMA, al recibir el PBU, actualizará su *Binding Cache*, y enviará un asentimiento del PBU, es decir un PBA.
- Se crearán las rutas necesarias para llevar a cabo la comunicación y el túnel entre el LMA y el MAG en caso de no existir con anterioridad.

Para llevarlo a cabo, en el LMA no será necesario realizar ningún tipo de modificación del código existente, ya que en el nuevo caso de que el MAG reciba otro mensaje en lugar de un RS, la señalización de éste será exactamente la misma a la realizada anteriormente, y el LMA recibirá el mismo tipo de mensajes, y deberá gestionarlos de igual manera.

En el caso de los puntos de acceso de nuestro entorno de pruebas (es decir, *routers*), el único cambio que se realizará en ellos será la instalación y ejecución del programa compilado mediante la técnica de *Cross Compiling*, descrito con detalle en el anexo E.

Es en los MAG dónde se realizan los cambios más significativos, ya que ahora deberán ser capaces de recibir otro tipo de mensajes, y actuar en consecuencia a ellos. Destacar que cuando un MAG recibe un mensaje del *router* con tipo 0 (de desconexión de un MN de su red), no emprenderá ninguna acción, debido a que según esta implementación la manera que tiene un MAG para eliminar un nodo de su BUL, es decir de su lista de nodos a los que gestiona la movilidad, es mediante el tiempo de vida de dicha entrada en la BUL.

El tiempo de vida se establece inicialmente a 60 segundos. El programa estima el tiempo necesario para llevar a cabo la señalización mediante mensajes y la creación de túneles y rutas, que en nuestro entorno de pruebas es de 3 segundos, por lo que 3 segundos antes de que expire la entrada de la BUL, lleva a cabo DAD, y en el caso de que el nodo se encuentre todavía en su red, actualizará el tiempo de vida de dicha entrada, nuevamente a 60 segundos,

y además, enviará un PBU al LMA indicando que el nodo continúa en su red y que por lo tanto debe actualizar el tiempo de vida de su *Binding Cache*.

Este método, aunque no sea el óptimo para gestionar las desconexiones de MNs, se considera apropiado. Por lo tanto, en vez de eliminar el MN del MAG cuando se recibe un mensaje procedente del *router*, y con tipo 0 (que será ignorado) simplemente se esperará a que venza el tiempo de vida de dicho nodo, y que se elimine de la misma forma que se realizaba anteriormente.

La manera más sencilla de añadir la funcionalidad para que el programa sea capaz de recibir los nuevos tipos de mensajes procedentes del *router* es creando un hilo de ejecución nuevo. Dicho hilo será independiente de todos los demás, siendo creado con el estado *detached*, ya que no será necesario en ningún momento que dicho hilo tenga que esperar que otro finalice para poder seguir ejecutándose. Tampoco utilizará ningún dato que pueda ser facilitado por otro hilo, simplemente utilizará el nuevo mensaje recibido y los ficheros de configuración.

Los demás datos necesarios para que se lleve a cabo esta funcionalidad, son introducidos inicialmente en el código, ya que son datos conocidos. Estos datos son:

- *Puerto*: por el cual se recibirán los mensajes enviados por el *router*. Éste ha de ser el mismo que se le indica al *router* cuando se procede a ejecutar el programa en él. Recordar, que en el *router* se especificaba un puerto de UDP, y que siempre se ha introducido el valor 1234.
- *Interfaz*: según la implementación utilizada es necesario saber el índice de la interfaz del MAG por la que éste recibe el mensaje del *router*. En este caso, el nombre de la interfaz en ambos ordenadores es “eth0_rename”, y coincide con el número 3.
- *Dirección MAG*: la dirección IPv6 de acceso local por la que el MAG recibe el mensaje del *router*. Coincide con la dirección correspondiente a la interfaz especificada. Este dato, aunque se introduce en el código, se podría obtener del fichero de configuración ya que corresponde con la variable “FixedMAGLinkLayerAddressOnAllAccessLink”. Esta variable en ambos MAG obtiene el mismo valor, ya que se forzó a que la dirección de la interfaz que une ambos MAG con su correspondiente punto de acceso sea la misma.
- *Dirección MN*: por último, también es introducida la dirección de acceso local del nodo móvil que ha entrado en la red de uno de los MAG. En este caso se introducirá la correspondiente con la interfaz “Wlan0” del MN, ya que será la que en una primera instancia se utilice para hacer pruebas.

El hilo creado llevará a cabo las mismas acciones que cuando se recibía un RS en el programa inicial. Por ello, lo primero que deberá ser capaz de hacer será recibir mensajes enviados por el *router*. Para ello, el hilo se encontrará siempre en un bucle infinito que esté

permanentemente a la espera de nuevos mensajes. Para ello, se creará un *socket pasivo* que escuche por la dirección del MAG y en el puerto especificado.

En el momento que llega un mensaje a dicho *socket*, entonces se llevan a cabo las acciones pertinentes. No se realiza ningún tipo de comprobación del mensaje recibido ya que se supone que siempre va a ser uno enviado por el punto de acceso y por lo tanto el formato de dicho mensaje será el esperado. Lo primero que se debe realizar es comprobar el tipo del mensaje que llega. En caso de ser 0, como se ha comentado anteriormente simplemente se ignora. Si por el contrario el mensaje es de tipo 1, entonces se invoca la función *mag_recv_event_attach*, que sustituirá a la función que se ejecutaba cuando se recibía un RS.

A la función invocada se le pasará por parámetro el otro dato que envía el punto de acceso en el mensaje, es decir la dirección HW del nodo que se ha conectado a su red, que corresponderá con el ID del MN, y por lo tanto ya no será necesario recuperar dicho dato del perfil del nodo. Al igual que la función a la que sustituye, ésta se encargará de recuperar el perfil del MN que se encuentra en el fichero de configuración, actualizar la BUL, en caso de que no se encuentre en ella, también asignará el correspondiente prefijo, crear el PBU y enviarlo, recibir el PBA, parsearlo y crear las rutas y el túnel necesario para que la comunicación entre el MN y el LMA se produzca de una manera satisfactoria.

A lo largo de su desarrollo, se encontraron ciertos problemas o comportamientos inesperados del mismo, obteniendo como resultado un funcionamiento erróneo del protocolo. Con el cual, en algunas ocasiones, al intentar simular un movimiento del MN, es decir, cambio de punto de acceso, la comunicación entre el MN y el LMA quedaba interrumpida. Observando las trazas obtenidas, tanto en el LMA como en el MAG, se encontraban ciertas anormalidades:

- **LMA:** según la traza, cuando el LMA recibía un nuevo PBU relativo a un MN al que él le gestionaba la movilidad, pero desde otro MAG distinto, cuando no tenía un comportamiento correcto, se quedaba en un bucle infinito buscando la entrada del nodo móvil en su *Binding Cache*. El problema residía en que nunca encontraba la entrada de dicho nodo en ella, y por lo tanto estaba continuamente buscándolo. La búsqueda que hace el LMA en la *Binding Cache* tiene como criterio de búsqueda el prefijo del nodo móvil, y cuando no se podía encontrar era porque el prefijo del nodo no era el que correspondía, era un híbrido entre dicho prefijo y su dirección MAC, y por lo tanto la búsqueda nunca se producía con éxito.

  Dado que el código del LMA no se modificó en ningún momento, se pensó que el error provendría de la generación del PBU en el MAG. Tras varias comprobaciones, y tras examinar los paquetes de señalización utilizando el *Wireshark*, no se detectó ningún fallo en los PBUs que originaba el MAG, ya que el valor de todos los campos era el esperado.

- **MAG:** en el caso del MAG se detectaron dos comportamientos anómalos, uno relacionado con la funcionalidad que se quería incluir, y otro en una funcionalidad

que se había dado por válida. Éste último está relacionado con lo comentado en este mismo apartado de cómo se da de baja un nodo en un MAG. Cómo ya se ha explicado, esto se realiza utilizando DAD. En el caso de que DAD devuelva éxito indicará que no existe un nodo con esa dirección en esa red, con lo cual el MN se habría desconectado. Si por el contrario da fallo, no se podría utilizar esa dirección porque ya está en uso, e indicará que el MN sigue en la red. Lo que sucedía con él, es que en algunos casos la implementación de DAD devolvía éxito cuando el MN aún seguía conectado a la red del MAG que lo ejecutaba, y además no habiendo recibido un evento del *router* de tipo 0.

El problema relacionado con la nueva funcionalidad, consistía, según lo observado en la traza del programa, en que una vez el MAG había creado o actualizado la entrada del MN en su BUL y enviado el PBU correspondiente, hacía una llamada a la búsqueda en la *Binding Cache*. La *Binding Cache* es exclusiva del LMA, en el caso del MAG dónde se almacena los datos de los MN es únicamente en la BUL. Se comenzaba a ejecutar otro hilo, que además en el caso concreto de ese hilo no debería ejecutar en el MAG, ya que corresponde a un tipo de acción únicamente asignada al rol del LMA. Con motivo de este error, se pensó que el fallo podría residir en un error en la gestión de los recursos compartidos por los diferentes hilos, como es la BUL. Por lo tanto se utilizó semáforos para no permitir a otro hilo acceder a ella mientras el hilo que se había creado la estaba modificando, controlando además, de una manera exhaustiva las veces que se accedía a ella para averiguar en qué momento se cambiaba de hilo. No se llegó a ninguna conclusión. Lo cual es lógico, porque en un primer momento no se utilizaron este tipo de variables de bloqueo, ya que el nuevo hilo sería el único que tuviese acceso a la BUL, pero se consideró necesaria esta comprobación por si a caso alguno de ellos la modificada o simplemente accedía a ella y se había obviado.

El último paso que se llevó a cabo para incluir correctamente esta funcionalidad al programa fue probar a introducir la nueva función implementada en el código original, es decir, en vez de llamar a la función que se llamaba originariamente cuando se recibía un RS, llamar a la nuestra, y viceversa, cambiar en el nuevo programa nuestra función por la que existía originariamente. Con ello se obtuvo:

- En el primer caso expuesto, cuando el programa que debía ser ejecutado en el *router*, no se estaba ejecutando, y por lo tanto para que funcionase se debía levantar la interfaz del MN, PMIPv6 funcionaba correctamente.
- En el segundo caso, que coincide cuando el programa del *router* estaba siendo ejecutado, y utilizando la función original, PMIPv6 no funcionaba siempre como se esperaba, obteniendo los mismos resultados que con nuestra función.

Cuando se habla de que el funcionamiento no era el esperado, es porque quedaba interrumpida durante un período de tiempo no admisible la comunicación entre el MN y el LMA (comprobada mediante el comando *ping6* desde el MN al LMA). No se llegó a averiguar

con exactitud cuándo sucedía. La primera vez que se iniciaba el protocolo, y cuando el MN se conectaba por primera vez a la red de un MAG, siempre se conseguía establecer correctamente la comunicación, pero si el MN se cambiada al otro MAG del entorno de pruebas antes de que expirase la entrada de la BUL en el primer MAG al que se conectó (en dicho caso si se conseguía el objetivo), no se llegaba a ejecutar correctamente. Esto se solucionaba si el MN volvía a conectarse por segunda vez, al último MAG al que se conectó. Entonces, el protocolo se volvía a ejecutar correctamente. Se continúan las pruebas, haciendo que el MN se conecte al otro MAG, y la comunicación se seguía manteniendo, durante un determinado número de cambios de punto de acceso variable. En uno de esos cambios, se perdía la comunicación, y nuevamente era solucionada si el MN se volvía a conectar al mismo MAG. No se pudo determinar ni el porqué ni la frecuencia de este comportamiento.

Al comprobar el mismo comportamiento tanto en la función que nosotros añadimos, como en la función original de la implementación mientras el programa del *router* se estaba ejecutando, se comprobó que lo que fallaba no era dicha función que se había añadido.

Debido a las dificultades encontradas, y añadiendo la complejidad de depurar un error de este tipo, dada su presunta aleatoriedad, hacen de esta implementación una aplicación muy inestable, con la cual no se consideró viable continuar trabajando con ella. Fue entonces cuando se decidió comenzar una nueva implementación de PMIPv6 más sencilla, y que sólo contemplase lo básico de él, incluida esta nueva funcionalidad. Con lo cual, se facilitará la depuración de cualquier tipo de inconveniente. Aunque no se implemente la funcionalidad completa, lo interesante será que al menos aporte una estabilidad que esta implementación anterior no aportaba, consiguiendo así, que la comunicación entre el MN y el LMA siempre se produzca de una manera satisfactoria.

H. Tabla t-STUDENT

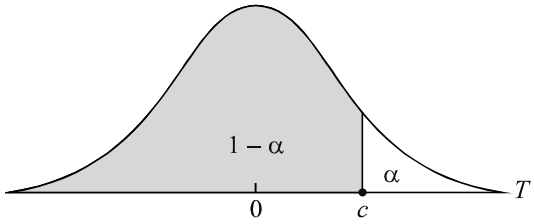


Tabla 10: t-STUDENTT

	1 – α							
r	0.75	0.80	0.85	0.90	0.95	0.975	0.99	0.995
1	1.000	1.376	1.963	3.078	6.314	12.706	31.821	63.657
2	0.816	1.061	1.386	1.886	2.920	4.303	6.965	9.925
3	0.765	0.978	1.250	1.638	2.353	3.182	4.541	5.841
4	0.741	0.941	1.190	1.533	2.132	2.776	3.747	4.604
5	0.727	0.920	1.156	1.476	2.015	2.571	3.365	4.032
6	0.718	0.906	1.134	1.440	1.943	2.447	3.143	3.707
7	0.711	0.896	1.119	1.415	1.895	2.365	2.998	3.499
8	0.706	0.889	1.108	1.397	1.860	2.306	2.896	3.355
9	0.703	0.883	1.100	1.383	1.833	2.262	2.821	3.250
10	0.700	0.879	1.093	1.372	1.812	2.228	2.764	3.169
11	0.697	0.876	1.088	1.363	1.796	2.201	2.718	3.106
12	0.695	0.873	1.083	1.356	1.782	2.179	2.681	3.055
13	0.694	0.870	1.079	1.350	1.771	2.160	2.650	3.012
14	0.692	0.868	1.076	1.345	1.761	2.145	2.624	2.977
15	0.691	0.866	1.074	1.341	1.753	2.131	2.602	2.947
16	0.690	0.865	1.071	1.337	1.746	2.120	2.583	2.921
17	0.689	0.863	1.069	1.333	1.740	2.110	2.567	2.898
18	0.688	0.862	1.067	1.330	1.734	2.101	2.552	2.878
19	0.688	0.861	1.066	1.328	1.729	2.093	2.539	2.861
20	0.687	0.860	1.064	1.325	1.725	2.086	2.528	2.845
21	0.686	0.859	1.063	1.323	1.721	2.080	2.518	2.831
22	0.686	0.858	1.061	1.321	1.717	2.074	2.508	2.819
23	0.685	0.858	1.060	1.319	1.714	2.069	2.500	2.807
24	0.685	0.857	1.059	1.318	1.711	2.064	2.492	2.797
25	0.684	0.856	1.058	1.316	1.708	2.060	2.485	2.787
26	0.684	0.856	1.058	1.315	1.706	2.056	2.479	2.779
27	0.684	0.855	1.057	1.314	1.703	2.052	2.473	2.771
28	0.683	0.855	1.056	1.313	1.701	2.048	2.467	2.763
29	0.683	0.854	1.055	1.311	1.699	2.045	2.462	2.756
30	0.683	0.854	1.055	1.310	1.697	2.042	2.457	2.750
40	0.681	0.851	1.050	1.303	1.684	2.021	2.423	2.704
60	0.679	0.848	1.046	1.296	1.671	2.000	2.390	2.660
120	0.677	0.845	1.041	1.289	1.658	1.980	2.358	2.617
∞	0.674	0.842	1.036	1.282	1.645	1.960	2.326	2.576

H. Listado de referencias

[1] R.A.E. (Real Academia Española)

[2] Hesham Soliman. *Mobile IPV6: Mobility in a wireless Internet.* Páginas 3-120. Addison-Wesley, 2004.

[3] S. Deering, Cisco. R. Hinden, Nokia. *Internet Protocol, version 6 (IPv6) Specification*. RFC 2460 (Standards Track), December 1998.

[4] D. Johnson, Rice University. C. Perkins, Nokia research center. J. Arkko, Ericsson. *Mobility Support in IPv6.* RFC 3775 (Standards Track), June 2004.

[5] S. Gundavelli, Ed. K. Leung, Cisco. V. Devarapalli, Wichorus. K. Chowdhury, Starent Networks. B. Patil, Nokia. *Proxy Mobile IPv6.* RFC 5213 (Proposed Standard), August 2008.

[6] S. Kent, K. Seo, BBN Technologies. *Security Architecture for the Internet Protocol.*RFC 4301 (Standards Track), December 2005.

[7] http://luanorma.blogspot.com/2006/02/compilar-kernel-al-modo-debian.html, visitada por última vez en Enero 2010.

[8] http://www.mogaal.com/articulos/kernel-a-la-debian.html , visitada por última vez en Enero 2010.

[9] http://packages.debian.org/, visitada por última vez en Abril 2010.

[10] http://www.escomoposlinux.org/, visitada por última vez en Mayo 2010.

[11] http://openwrt.org/, visitada por última vez en Julio 2010.

[12] V. Devarapalli, Nokia. R. Wakikawa, Keio University. A. Petrescu, Motorola. P. Thubert, Cisco systems. *Network Mobility (NEMO) Basic Support Protocol.* RFC 3963 (Proposed Standard), January 2005.

[13] A. Conta, Transwitch. S. Deering, Cisco systems. M. Gupta, Ed. Tropos Networks. *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification.*RFC 4443 (Draft Standard), March 2006.

[14] T. Narten, IBM. E. Nordmark, Sun Microsystems. W, Simpson, Daydreamer. H. Soliman, Elevate Technologies. *Neighbor Discovery for IP version 6 (IPv6).* RFC 4861 (Draft Standard), September 2007.

[15] S. Chakrabarti, E. Nordmark, Sun Microsystems. *Extensions to sockets API for Mobile IPv6*. RFC 4584 (Informational), July 2006.

[16] W. Stevens, M. Thomas, Consultant. E. Nordmark, Sun Microsystems. T. Jinmei, Toshiba. *Advanced Sockests Application Program Interface (API) for IPv6*. RFC 3542 (Informational), May 2003.

[17] Peña, Daniel. *Fundamentos de estadística.* Alianza Universidad, 2001.

[18] http://linuxkillwin.blogspot.com/2010/02/aircrack-ng-ubuntu-910-modo-monitor.html visitada por última vez en Diciembre 2010.

[19] Sarkar, Subir Kumar. *Ad Hoc Mobile Wireless Networks: principles, protocols and applications.* Auerbach, 2008.